コンテナセキュリティ

センサー導入ガイド

バージョン 1.28

2023 年 8 月 14 日

Qualys, Inc.
919 E Hillsdale Blvd
4th Floor
Foster City, CA 94404
1 (650) 801 6100

目次

このガイドについて

Qualys Container Security へようこそ。Qualys Cloud Security Platform を使用して、イメージ、コンテナ、Docker ホストなどのコンテナ環境を保護するための Qualys ソリューションについて理解を深めるお手伝いをします。

Qualys について

Qualys, Inc.(NASDAQ:QLYS)は、クラウドベースのセキュリティおよびコンプライアンスソリューションのパイオニアであり、リーディングプロバイダーです。Qualys Cloud Platform とその統合アプリは、重要なセキュリティインテリジェンスをオンデマンドで提供し、IT システムと Web アプリケーションの監査、コンプライアンス、保護の全範囲を自動化することで、企業がセキュリティ運用を簡素化し、コンプライアンスのコストを削減するのに役立ちます。

1999 年に設立された Qualys は、アクセンチュア、BT、コグニザント・テクノロジー・ソリューションズ、ドイツテレコム、富士通、HCL、HP Enterprise、IBM、インフォシス、NTT、Optiv、SecureWorks、タタ・コミュニケーションズ、ベライゾン、ウィプロなどの大手マネージド・サービス・プロバイダーやコンサルティング組織と戦略的パートナーシップを結んでいます。また、Cloud Security Alliance(CSA)の創設メンバーでもあります。 詳細については、www.qualys.com をご覧ください。

Qualys サポート

Qualys は、徹底したサポートを提供することをお約束します。Qualys は、オンラインドキュメント、電話によるヘルプ、および直接の電子メールサポートを通じて、お客様の質問に可能な限り迅速に回答できるようにします。週 7 日、24 時間体制でサポートします。
www.qualys.com/support/ でオンラインサポート情報にアクセスし ます。

コンテナセキュリティドキュメントについて

このドキュメントでは、MAC、CoreOS、およびさまざまなオーケストレーターとクラウド環境へのセンサーの導入について説明します。

コンテナセキュリティの UI と API の使用については、以下を参照してください。

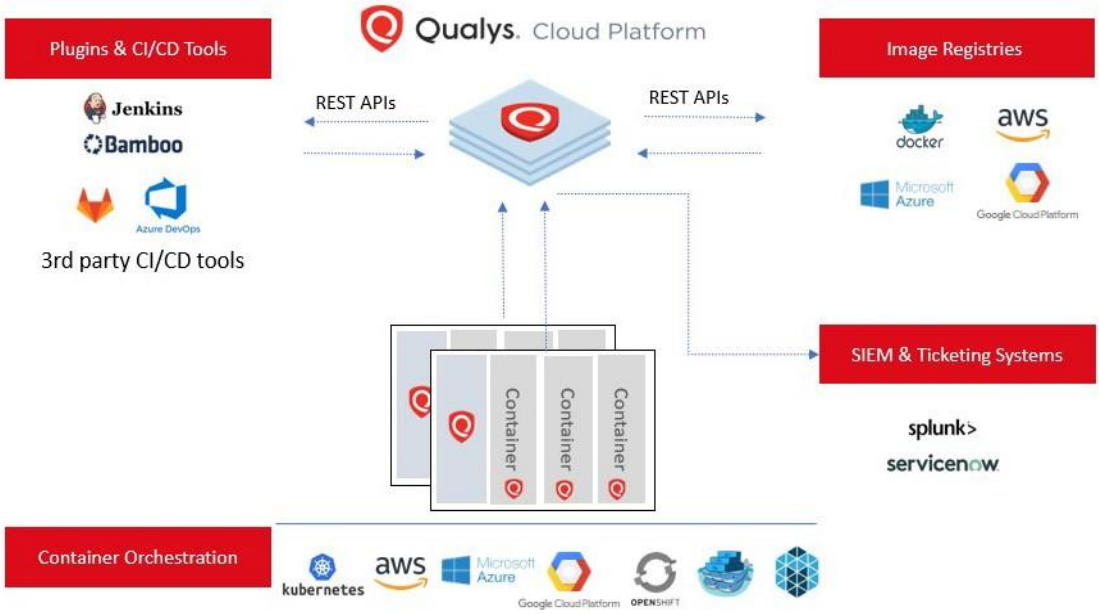
- Qualys Container Security User Guide
- Qualys Container Runtime Security User Guide
- Qualys Container Security API Guide
- Qualys Container Runtime Security API Guide

CI/CD 環境でのセンサーの展開については、以下を参照してください。

- Qualys Container Scanning Connector for Jenkins
- Qualys Container Scanning Connector for Bamboo
- Qualys Container Scanning Connector for Azure DevOps

コンテナセキュリティの概要

Qualys Container Security は、コンテナ環境の検出、追跡、および継続的な保護を提供します。これにより、DevOps パイプライン内のイメージとコンテナの脆弱性管理とポリシーコンプライアンス、およびクラウド環境とオンプレミス環境にわたるデプロイに対応します。



このバージョンでは、Qualys Container Security は以下をサポートします。

- コンテナ環境の検出、インベントリ、およびほぼリアルタイムの追跡
- イメージとコンテナの脆弱性分析
- レジストリの脆弱性分析
- イメージとコンテナのコンプライアンス評価
- API を使用した CI/CD パイプラインとの統合 (DevOps フロー)
- 「Container Sensor」を使用 - Docker イメージとして配布されるネイティブコンテナサポートを提供

Qualys コンテナセンサー

Qualys のセンサーは、Docker 環境をネイティブにサポートするように設計されています。センサーは Docker イメージとしてパッケージ化され、提供されます。イメージをダウンロードし、ホスト上の他のアプリケーションコンテナと一緒にコンテナとしてデプロイします。

センサーは Docker ベースで、データセンターのホストや AWS ECS などのクラウド環境にデプロイできます。現在、センサーは Linux オペレーティング システムでのみサポートされており、バージョン 1.12 以降の docker デーモンが使用可能である必要があります。

Docker ベースであるため、センサーは他のアプリケーションコンテナと同様に、Kubernetes、Mesos、Docker Swarm などのオーケストレーションツール環境にデプロイできます。

インストール時に、センサーはデプロイされたホスト上のイメージとコンテナを自動的に検出し、それらの脆弱性分析を提供し、さらにホスト上の Docker 関連イベントを監視および報告します。センサーは、レジストリを一覧表示してスキャンし、脆弱なイメージを探します。センサーはコンプライアンス評価も実行します。センサー コンテナは、非特権モードで実行されます。ファイルを保存およびキャッシュするための永続ストレージが必要です。

現在、センサーはイメージとコンテナのみをスキャンします。ホストで脆弱性ポスチャを取得するには、Qualys Cloud Agents または Qualys Virtual Scanner Appliance によるスキャンが必要です。

センサーモード

センサーは、1 つのコンテナのホスト/クラスター ノードに 1 つのモードでのみデプロイできます。

全般

全般（General）モード・センサーは、コンテナ・ノード/ホストにインストールされます。実行中のコンテナとローカルにキャッシュされたイメージの脆弱性とコンプライアンスの評価を提供します。全般センサーは、インスタンス化されたコンテナやプルされたイメージなどのコンテナイベントに基づいて、需要主導の評価を実行します。オンデマンド スキャンまたはスケジュールされたスキャンの評価はありません。センサーは、コンテナ環境の変化にリアルタイムで反応します。全般モード・センサーは、レジストリー・センサーまたは CICD センサーとは別にデプロイする必要があります。

レジストリ

レジストリモードでは、レジストリに格納されているイメージのインベントリと脆弱性の評価が提供されます。レジストリ モードのセンサーは、センサーが展開されているホスト上のイメージまたはコンテナのインベントリを作成したり、脆弱性評価を実行したりしません。レジストリモードのセンサーには、レジストリ URL へのネットワーク アクセスが必要です。レジストリ モード センサーは、レジストリを自動的に検出しません。インベントリおよび評価されたイメージは、レジストリ コネクタ スキャン ジョブによってスコープが設定されます。これらのスキャン ジョブは、自動 (スケジュール) またはオンデマンドのいずれかです。Container Security UI にログインして、レジストリコネクタとスキャンジョブを設定します。ガイダンスについては、オンライン・ヘルプを参照してください。レジストリー・モード・センサーは、一般センサーまたは CICD センサーとは別にデプロイする必要があります。

CICD

CICD モードは、CI パイプライン ワーカーで実行されているセンサー用です。これは、特定のイベントに基づく需要主導の評価です。CICD モードのセンサーは、ホスト/ノードで実行されている他のイメージまたはコンテナのインベントリや評価を行いません。CICD モードのセンサーは、特別にタグ付けされたイメージに対して脆弱性評価を実行し、評価結果は、CI パイプライン評価専用のより高速な SLA で優先処理キューに入れられます。CICD センサーは、一般センサーまたはレジストリー・センサーとは別にデプロイする必要があります。

Container Security はどのようなデータを収集しますか?

Qualys コンテナ セキュリティ センサーは、環境内のイメージとコンテナに関する次の情報を取得します。

すべてのコンテナを一覧表示する docker ps などのコマンドからの環境内のイメージとコンテナのインベントリ。

docker inspect や docker オブジェクトに関する低レベルの情報を取得する docker info などのコマンドからのイメージとコンテナに関するメタデータ情報。

作成、開始、強制終了、プッシュ、プルなどの Docker イベントに関する Docker ホストからのイメージとコンテナに関するイベント情報。

イメージとコンテナで見つかった脆弱性。これは、イメージとコンテナの脆弱性情報を識別するために実行される脆弱性管理マニフェストの出力です。これは主に、ソフトウェアパッケージのリスト、実行中のサービス、ポートなどです。たとえば、パッケージ マネージャーは rpm -qa、npm のように出力します。これは、さまざまな Linux ディストリビューション(CentOS、Ubuntu、CoreOS など)と、Python、NodeJS、Ruby などのイメージでサポートされています。

OCI 準拠のイメージ、実行中のコンテナのコンプライアンス構成。CIS Docker ベンチマークのコントロールのサブセットをサポートしており、コンテナとコンテナイメージの実行に適用できます。お客様は、実行中のコンテナとイメージの構成リスクを評価し、Qualys の調査結果に基づいて適切に修正できます。コンテナやイメージのコンプライアンススキャンは、お客様に対して透過的であり、脆弱性スキャン機能と同様にリアルタイムのクラウドネイティブな方法で機能します。

はじめに

Qualys サブスクリプションとモジュールが必要

アカウントで「コンテナセキュリティ」(CS)モジュールを有効にする必要があります。また、コンテナを実行するホストの脆弱性を取得するには、Scanner Appliance または Cloud Agent を使用して脆弱性管理(VM)を有効にする必要があります。

システムサポート

コンテナセキュリティセンサーは、Docker バージョン 1.12 以降のオペレーティングシステムで実行できます。センサーは、次のシステムで検証済みです。

- CentOS Linux 7.3, 7.4, 7.5, 7.6, 7.7
- CoreOS 1855.4.0
- Debian Linux 8 (Jessie)
- Fedora Release 28
- Kali Linux
- Mac OS 10.13
- Red Hat Enterprise Linux 7.4
- Red Hat Enterprise Linux Atomic Host 7.5, 7.7
- Ubuntu 14.04, 16.04, 18.04, 20.04

コンテナセキュリティセンサーは、次のオペレーティングシステムに基づいてコンテナイメージをスキャンできます。

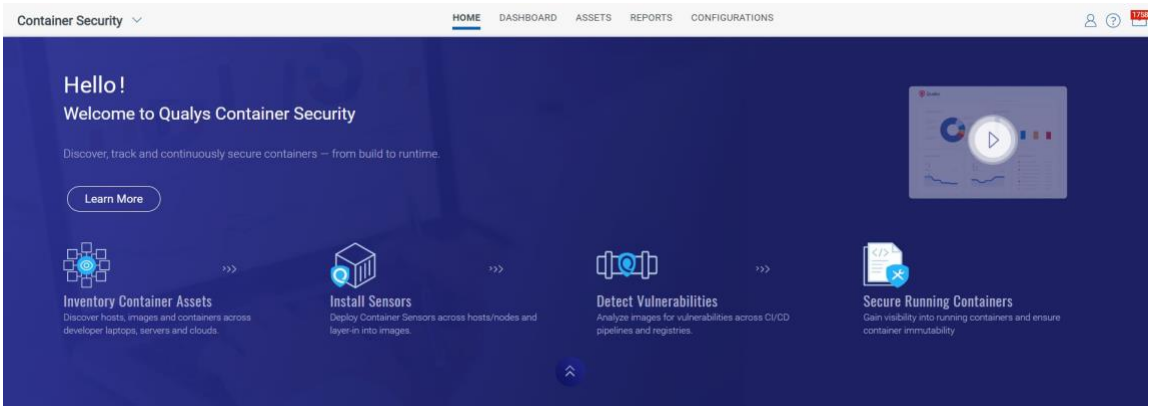
- AlmaLinux
- Alpine Linux
- Amazon Linux
- CBL-Mariner Linux
- CentOS
- Debian
- Fedora
- Google Distroless (Debian Linux)
- OpenSUSE
- Oracle Enterprise Linux
- Red Hat Enterprise Linux Server
- SUSE Linux Enterprise Server
- Ubuntu

Container Sensor のデプロイ

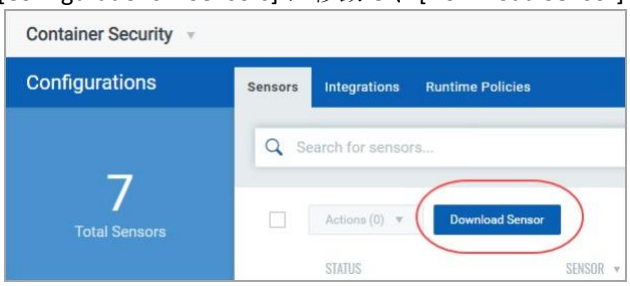
コンテナセキュリティセンサーは、次のいずれかの方法でインストールできます。

- Qualys Cloud Platform からセンサー tar ファイルをダウンロードし、ホストにインストールします。
- Docker Hub からセンサーをインストールします。「Docker Hub からのセンサーのインストール」を参照してください。

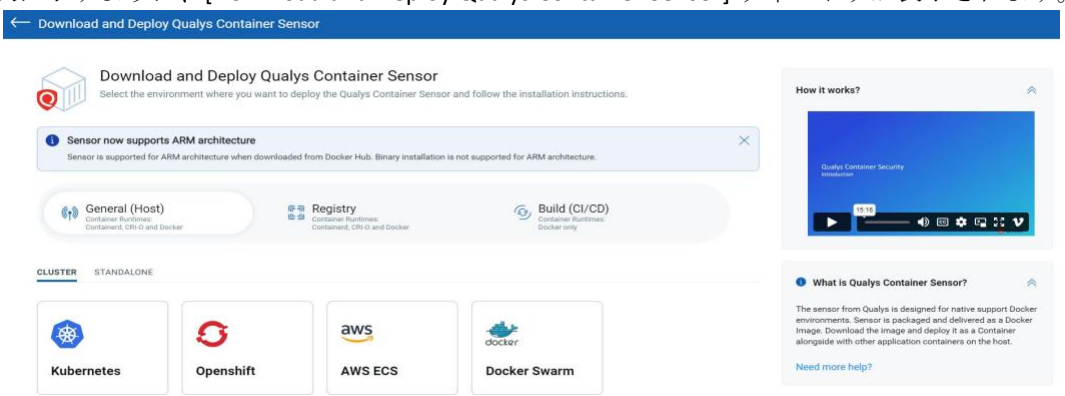
Qualys Cloud Platform からセンサーをダウンロードするには、ユーザ クレデンシャルを使用して Qualys ポータルにログインします。モジュールピッカーから [Container Security] を選択します。初めてのユーザーは、ホームページに直接アクセスします。



[Configurations > Sensors] に移動し、[Download Sensor] をクリックします。



次に示すように、[Download and Deploy Qualys Container Sensor] ウィンドウが表示されます。



デプロイするセンサーの種類を選択します。

重要: センサーの展開は、1 つのホスト/ノード上の 1 つのモードに 1 つのセンサーです。複数のセンサーまたは複数のセンサーを別のモードで展開することはサポートされていません。

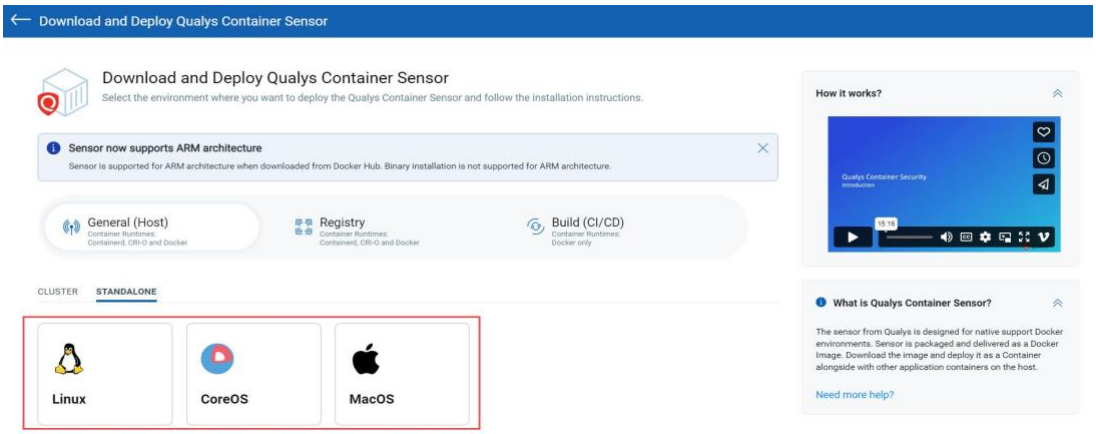
センサーモード:

一般 (ホスト) センサー: センサーが実行されているホスト上のイメージとコンテナをスキャンします。一般センサーは、レジストリまたは CI/CD のパラメーターが指定されていない場合、デフォルトでインストールされます。

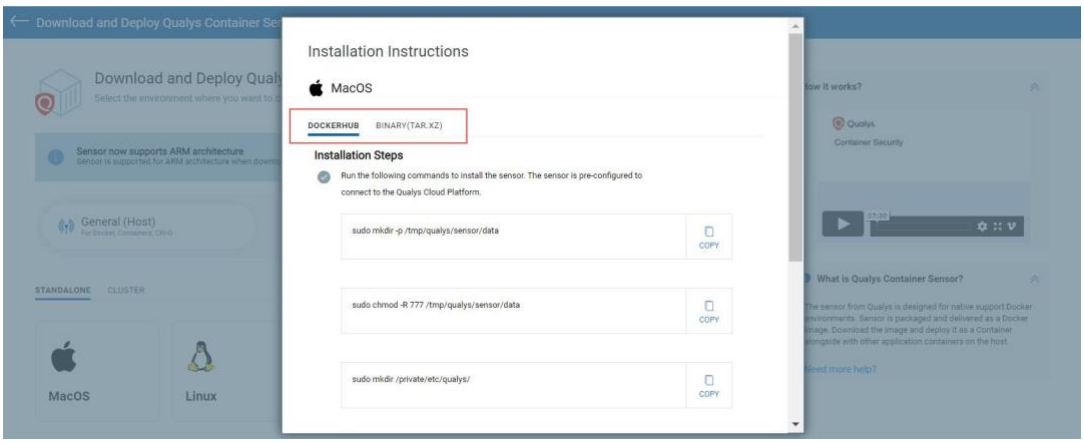
Registry センサー:レジストリ(パブリック/プライベート)内のイメージをスキャンします。レジストリの場合は、インストールコマンドに**--registry-sensor** または**-r** を追加する必要があります。

ビルド (CI/CD) センサー: CI/CD パイプライン (Jenkins/Bamboo) でイメージをスキャンします。CI/CD の場合は、install コマンドに **--cicd-deployed-sensor** または **-c** を追加する必要があります。

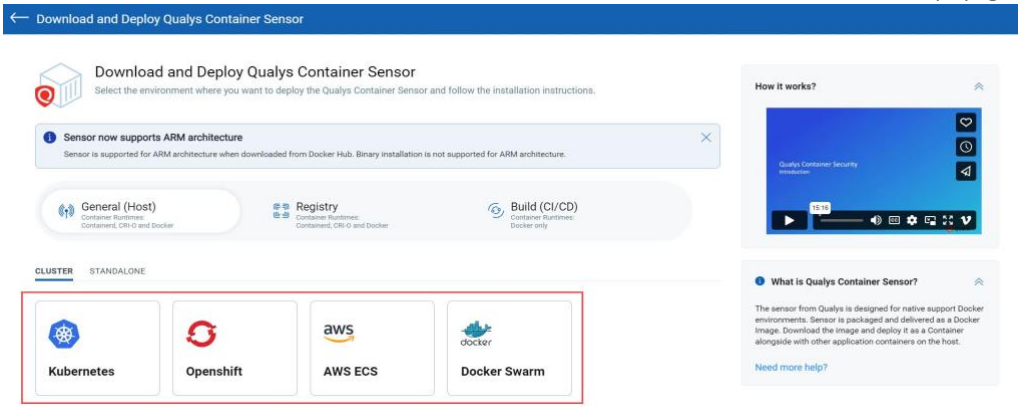
スタンドアロン ホストにデプロイするには、ホストのオペレーティング システム (MacOS、Linux、または CoreOS) を選択します。



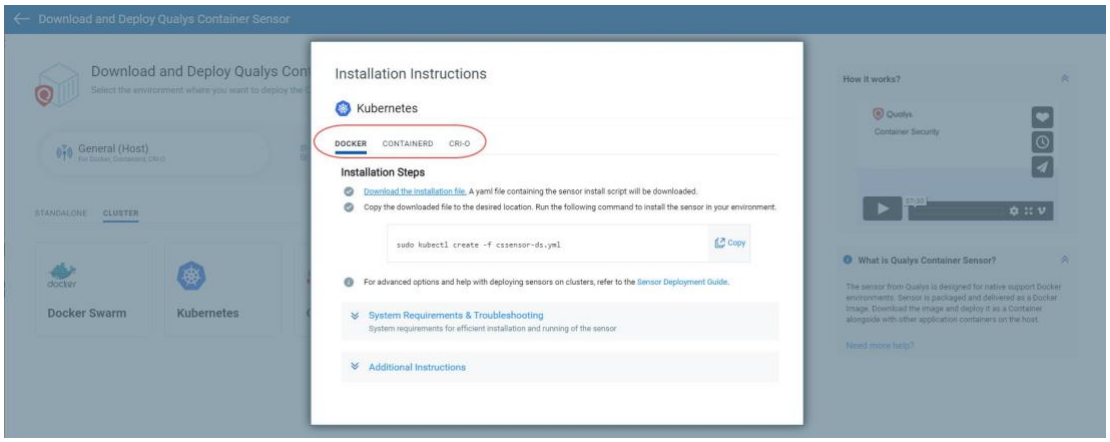
表示されるウィンドウで、DOCKERHUB または BINARY(TAR.XZ) を使用して、センサーの取り付け方法を指定します。次に、画面の手順に従います。Tar の場合は、tar ファイルをダウンロードし、画面上でインストール コマンドを実行します。Docker Hub の場合は、画面上で docker コマンドを実行します。



クラスターにデプロイするには、まずクラスター オプション (Kubernetes、Openshift、AWS ECS、または Docker Swarm) から選択します。



表示されるウィンドウで、ランタイムを選択します。Kubernetes にデプロイされている一般的なセンサーの次の例では、DOCKER、CONTAINERD、および CRI-O ランタイム オプションが表示されます。選択したら、画面の手順に従います。インストール yaml ファイルには、アクティベーション ID、カスタマー ID、および POD URL が事前に入力されています。



センサーを取り付けるためのシステム要件を必ずメモしてください。センサーには、ホスト上に最低 1 GB の永続ストレージが必要です。

Installsensor.sh スクリプトのコマンドラインパラメータ

「installsensor.sh」スクリプトのコマンドラインパラメータの概要を簡単に説明します。

注:デフォルト値を持つパラメータはごくわずかです。デフォルト値は、センサーの取り付け時に変更できます。ただし、一度設定したデフォルト値(LogLevel など)は、設定の更新によって上書きされる可能性があります。センサーのインストール後に既定値を変更する場合は、新しい値で "installsensor.sh" スクリプトを再実行する必要があります。

Parameter	Description
ActivationId	(Required) The Activation Id for the container sensor, autogenerated based on your subscription.
CustomerId	(Required) The Qualys subscription's customer Id, autogenerated based on your subscription.
--cicd-deployed-sensor or -c	(Optional) Run the sensor in a CI/CD environment. This allows you to scan images on CI/CD pipeline (Jenkins, Bamboo).
--registry-sensor or -r	(Optional) Run the sensor to list and scan registry assets. This allows you to scan images in a public or private registry.
Storage	(Optional) Directory where the sensor would store the files. Default value: /usr/local/qualys/sensor/data. Create this directory if not already available or specify a custom directory location.
--sensor-without-persistent-storage	(Optional) Use this option to run the sensor without using persistent storage on the host. Note: The sensor should be run either with the "--sensorwithout-persistent-storage" option OR with the "--read-only" option and not with both options enabled together. To install the sensor without persistent storage, exclude the "Storage" option, and include the "--sensor-without-persistent-storage" option in the installer script. We recommend you use the "--enable-console-logs" option along with "--sensorwithout-persistent-storage" to preserve the logs as data is not available on host but stored at the /usr/local/qualys/qpa/data folder relative to the Sensor. As the sensor is running with "--sensor-without-persistent-storage", upon auto-update the updated sensor is a completely new instance of sensor container hence data from the old sensor is not available in the new sensor. Thus, the new sensor rescans existing already scanned assets.

Parameter	Description
--read-only	(Optional) Use this option to run the sensor in read-only mode. In this mode the sensor uses persistent storage on the host. Note: The sensor should be run either with the "--sensorwithout-persistent-storage" option OR with the "--read-only" option and not with both options enabled together.
ConcurrentScan	(Optional) Number of docker/registry asset scans to run in parallel. A valid range is between 1-20. Default value is 4.
CpuShares	(Optional) Define CPU shares for the sensor container. A valid value is a non-zero, positive integer other than 1024.
CpuUsageLimit	(Optional) CPU usage limit in percentage for sensor. A valid range is between 0-100. Default is 0.2, i.e. 20% per core on the host. The installsensor script has intelligence to find the number of CPU cores present on the host and apply the CPU limit based on the CpuUsageLimit input value and number of CPU cores available. For example, when CpuUsageLimit=30, it's considered as 30% CPU of overall CPU capacity of the host. If the host has 8 CPU cores, the total CPU limit applied to sensor container would be 0.30 * 8 = 2.4 CPU cores.
--disable-auto-update	(Optional) Do not let sensor update itself automatically.
--disableImageScan	(Optional) This parameter should be passed if you want to disable image scans for General Sensor. Images will not be scanned by sensors deployed with this option. This is available for General sensor type only, and is available for all Runtimes (Docker, CRI-O and Containerd).
ScanningPolicy	(Optional) Specifies the scanning policy, which allows you to select the suitable scan type as per your requirement. The available values are: - `DynamicScanningOnly`: performs only dynamic scanning. - `StaticScanningOnly`: performs only static scanning. - `DynamicWithStaticScanningAsFallback`: performs static scanning as a fallback to dynamic scanning for images without shell.
--disable-log4j-scanning	(Optional) This parameter should be passed if you want to disable log4j vulnerability scanning for container images. See Log4j vulnerability scanning.
--disable-log4j-staticdetection	(Optional) This parameter should be passed if you want to disable log4j static detection for dynamic/static image scans. See Static log4j detection.

Parameter	Description
DockerHost	(Optional) The address on which the docker daemon is configured to listen. This option is mandatory if DOCKER_TLS_VERIFY=1 is defined. DockerHost format: <Docker daemon host's IPv4 address, or FQDN, or hostname>:<port#>
DockerSocketDirectory	(Optional) Docker socket directory path. The default value is Default: /var/run
DOCKER_TLS_VERIFY	(Optional) This parameter enables the TLS authentication. The value should be 0 or 1. Note: If DOCKER_TLS_VERIFY=1 is defined, then ensure that the provided IPv4 address or FQDN or hostname in DockerHost matches either the CN or the Alternative Subject Name in the docker server certificate. Note: By enabling sensor communication with docker daemon over TLS, customer can restrict the sensor's access to docker socket by using docker authorization plugin.
TLS_CERT_PATH	(Optional) Provide client certificate directory path. This is mandatory if DOCKER_TLS_VERIFY=1 is defined. tlscacert=<Name of CA (default "ca.pem")> tlscert=<Name of TLS certificate file (default "cert.pem")> tlskey=<Name of TLS key file (default "key.pem")> Note: If any of the CA certificate, client certificate, or client private key have default file names such as ca.pem, cert.pem, key.pem respectively they can be omitted.
--enable-console-logs	(Optional) Print logs on console. These logs can be retrieved using the docker logs command.
HostIdSearchDir	(Optional) Directory to map the marker file created by Qualys Agent or Scanner appliance on the host, update if modified. Default value is /etc/qualys
ImageFile	(Optional) Location of the Sensor ImageFile. This defaults to the local directory.
LogFilePurgeCount	(Optional) Integer value that specifies the maximum number of archived log files. Default value is 5.
LogFileSize	(Optional) Configuration to set the maximum size per log file for sensor in bytes. Accepts "<digit><K/M/>" where K is kilobytes and M is megabytes. For example, specify "10" for 10 bytes, "10K" for 10 kilobytes, "10M" for 10 megabytes. Default value is "10M".

Parameter	Description
LogLevel	(Optional) Configuration to set the logging level for sensor, accepts 0 to 5. Default value is 3 (Information).
--mask-env-variable	(Optional) Use this parameter to mask environment variables for images and containers. The environment variables will be masked/removed in sensor logs and in the Container Security UI.
MemoryUsageLimit	(Optional) Define the memory usage limit for the sensor container. The value should be formatted as <digit><unit> where unit can be any of the following: b (bytes), k (kilobytes), m (megabytes), g (gigabytes). The recommended value is 500m for 500 megabytes.
--optimize-image-scans	(Optional) This parameter should be passed if you want to optimize Image scans for General Sensor. This is available for General sensor type only. By default, the sensor scans every image that it detects on the host. This results in redundant scanning of images. When you install the General sensor with "--optimize-image-scans", the sensor will communicate with the Qualys Cloud Platform and perform informed scans to avoid redundant image scans. The sensor will determine if the images present on the host are already scanned by other sensors for the same manifest and version and will not scan those images again.
PidLimit	(Optional) Define the PID limit for the sensor container. The value provided must be a positive integer.
Proxy	(Optional) IPv4/IPv6 address or FQDN of the proxy server.
ProxyCertFile	(Optional) Proxy certificate file path. ProxyCertFile is applicable only if Proxy has valid certificate file. If this option is not provided, then Sensor will try to connect to the server with given https Proxy settings only. If only ProxyCertFile is provided without Proxy then Sensor would simply ignore the ProxyCertFile and it would try to connect to the server without any https proxy settings.
--silent or -s	(Optional) Run installsensor.sh in non-interactive mode.

SCA スキャンのオプション・パラメーター

以下のパラメーターは、サブスクリプションで SCA スキャン機能が有効になっている場合はオプションです。詳しくは、SCA スキャンを参照してください。

パラメーター	説明
--perform-sca-scan	(サブスクリプションで SCA スキャンが有効になっている場合のオプション) デフォルトでは、SCA スキャンは実行されません。このパラメーターを使用して、コンテナー・イメージの SCA スキャンを使用可能にします。指定すると、SCA スキャンは標準の脆弱性スキャン (静的または動的) の後に実行されます。SCA スキャンは、脆弱性スキャンが成功しない場合でも試行されます。
--disallow-internet-accessfor-sca	(--perform-sca-scan が指定されている場合のオプション)デフォルトでは、SCA スキャンではインターネット・アクセスが使用可能になっており、オンライン・モードでは、オフライン・モードで実行されるスキャンと比較してパッケージ・コレクションが正確であるため、SCA スキャンはオンライン・モードで実行されます。このパラメーターを使用して、SCA スキャンのインターネット・アクセスを使用不可にし、代わりにオフライン・モードでスキャンを実行します。 注: SCA スキャンはオンライン・モードで実行することをお勧めします。Java のソフトウェア・パッケージ列挙の品質は、SCA スキャンがオフライン・モードで実行されると大幅に低下します。正確なパッケージ検出のために、リモートの Maven リポジトリを参照する必要がある場合があります。これは、イメージの脆弱性ポスチャの精度に影響を与える可能性があります。センサーは URL "http://search.maven.org/" に到達できる必要があります。
SCAScanTimeoutInSeconds ={value}	(--perform-sca-scan が指定されている場合のオプション)デフォルトの SCA スキャン・コマンド・タイムアウトは 5 分 (300 秒) です。このパラメーターを使用して、デフォルトのタイムアウトを秒単位で指定した新しい値で上書きします。例えば、大きなコンテナー・イメージをスキャンするときには、SCA スキャンが完了する時間を確保するために、SCA スキャンのタイムアウトを増やす必要がある場合があります。 **注**: このパラメーターは、シークレット検出のタイムアウトを指定するためにも使用されます。

Docker Hub

Docker Hub からセンサーをインストールする方法については、以下を参照してください。

Installing the sensor from Docker Hub

CI/CD Environments

CI/CD 環境でのセンサーの展開については、以下を参照してください。

Qualys Container Scanning Connector for Jenkins

Qualys Container Scanning Connector for Bamboo

Qualys Container Scanning Connector for Azure DevOps

注: ホストは、HTTPS ポート 443 を介して Qualys Cloud Platform (または Qualys Private Cloud Platform) に到達できる必要があります。ホストがアクセスする必要がある Qualys プラットフォーム(POD URL)を参照してください。

Installsensor.sh コマンドを使用して Docker の CIS ベンチマークに準拠する方法

Qualys Container Security は、センサー イメージの Docker の CIS ベンチマークに準拠しています。Docker の CIS ベンチマークに準拠する方法で Sensor イメージを使用する方法については、Docker の CIS ベンチマークへの準拠を参照してください。準拠した方法でセンサーを操作できるように、いくつかのコントロールの手順を提供しています。

プロキシのサポート

インストール スクリプトは、プロキシ構成を要求します。IP アドレス/FQDN とポート番号、およびプロキシ証明書ファイルのパスを指定する必要があります。例えば :

```
Do you want connection via Proxy [y/N]: y
Enter Https Proxy settings [<IP Address>:<Port #>]: 10.xxx.xx.xx:3xxx
Enter Https Proxy certificate file path:
/etc/qualys/cloudagent/cert/ca-bundle.crt
```

プロキシ サーバは、HTTPS ポート 443 経由で Qualys Cloud Platform (または Qualys プライベート クラウド プラットフォーム) へのアクセスを提供する必要があります。ホストがアクセスする必要がある Qualys プラットフォーム(POD URL)を参照してください。

ホストがアクセスする必要がある Qualys プラットフォーム(POD URL)

使用する Qualys URL は、アカウントが配置されている Qualys プラットフォームによって異なります。

Qualys プラットフォームを特定し、Container Security Server の URL を取得するには、ここをクリックしてください。

POD URL 値

プラットフォームの「コンテナセキュリティサーバ URL」は、センサーをデプロイするときに、コンテナセキュリティセンサーコマンドおよび設定 yaml ファイルで POD_URL 変数に指定する必要がある URL です。

センサーネットワーク構成

センサーは、Qualys との通信に必要な Qualys URL とサブスクリプションの詳細で事前設定されています。センサーが Qualys と通信するには、ネットワーク設定とファイアウォールがポート 443 経由で Qualys ドメインにアクセスできるようにする必要があります。

センサーが正常にインストールされると、センサーは Container Security UI の [構成] > [センサー] に一覧表示され、バージョン、ステータスなどを確認したり、詳細にアクセスしたりできます。さらに、[Configurations > Sensors] の下のリンクからセンサーをダウンロードできます。

コンテナイメージの静的スキャン

静的スキャンは、スタンドアロンの Docker ホストでのデプロイと、Docker ランタイムと Containerd ランタイムを使用した Kubernetes でのデプロイでサポートされています。

センサーは、コンテナ イメージにシェルがない場合に、現在の動的スキャンへのフォールバック メカニズムとして、コンテナ イメージの静的スキャンを実行します。静的スキャンは、シェルのない Google ディストリビューションイメージに対しても実行されます。静的スキャンは、コンテナまたはシェルを持つコンテナイメージでは実行されません。

静的スキャンでは、インストールされているソフトウェアのリストをコンテナイメージファイルシステムから収集して、コンテナイメージの脆弱性を検出します。インストールされているソフトウェアの一覧は、パッケージ マネージャーのメタデータ ファイルから取得されます。サポートされているパッケージマネージャーは、RPM、DPKG、および Alpine です。

センサーが実行されているホストにシェルのない大きなイメージがある場合、ディスク容量の要件が最小要件の 1 GB を超える可能性があります。

Log4j の脆弱性スキャン

センサーは、Log4j リモート コード実行 (RCE) 脆弱性 QID を検出できます。コンテナイメージと実行中のコンテナ上の脆弱な log4j パッケージの存在を検出できます。センサーは、コンテナ イメージの log4j の脆弱性を検出するためにファイル システム検索を自動的に実行し、パフォーマンスに影響を与える可能性があります。log4j 脆弱性スキャンを無効にするには、**"installsensor.sh"** スクリプトのコマンドライン パラメーターとして **--disable-log4j-scanning** を指定するか、センサーのデプロイ時にコマンドまたは args パラメーターとして指定します。

静的 log4j 検出

静的 log4j 検出は、動的/静的イメージ スキャンに対して既定で有効になっています。静的な log4j 検出は、各画像レイヤーに対して log4j 検出コマンドを実行し、結果をマージすることで実装されます。パラメータ **--disable-log4j-static-detection** を使用して、動的/静的イメージスキャンの静的 log4j 検出を無効にするオプションがあります。

静的スキャンの場合、**--disable-log4jstatic-detection** パラメーターが指定されていない限り、静的 log4j 検出は常に呼び出されます。log4j コマンドは、VM マニフェストから読み取られます。

動的スキャンの場合、静的 log4j 検出は、次の条件が当てはまる場合にのみ呼び出されます。

- パラメーター **--disable-log4j-scanning** は使用されません。
- パラメーター **--disable-log4j-static-detection** は使用されません。

- プライマリ log4j 検出コマンドは、log4j データポイントの収集に失敗しました。

静的 log4j 検出を無効にするには、**"installsensor.sh"** スクリプトのコマンド ライン パラメーターとして **--disable-log4j-static-detection** を指定するか、センサーのデプロイ時にコマンドまたは args パラメーターとして指定します。

SCA スキャン

Qualys Container Security Sensor は、コンテナイメージのソフトウェアコンポジション分析(SCA)スキャンをサポートしています。SCA スキャンは、コンテナイメージに存在する、インストールされているオープンソースソフトウェアとライブラリ、および関連する脆弱性を検出します。

コンテナイメージのセキュリティ体制を評価する際には、イメージに存在するすべてのソフトウェアパッケージを特定することが重要です。SCA スキャンは、イメージ内のプログラミング言語ベースのソフトウェア・パッケージを識別するために使用できます。さらに、各画像レイヤーのメタデータ情報も提供されます。SCA スキャンは、Java、Python、Go、Node.js、.NET、PHP、Ruby、および Rust のプログラミング言語のパッケージを検出します。

SCA スキャンは、すべてのセンサー タイプ (一般、レジストリ、CI/CD) でサポートされています。これは、Docker、ContainerD、および CRI-O ランタイムでサポートされています。SCA スキャンは、コンテナ・イメージをスキャンする場合にのみサポートされます。SCA スキャンは Mac OS ではサポートされていません。

注: CRI-O ランタイムで SCA をサポートするには、特権権限でセンサーを起動する必要があります。これを行うには、「cssensor-crio-ds.yml」ファイルで、次のパラメーターを true に設定する必要があります。

```
securityContext:
privileged: true
```

前提条件

- サブスクリプションで SCA スキャン機能が有効になっている必要があります。Qualys サポートに連絡して、この機能を有効にしてください。
- センサー バージョン 1.19 以降。
- パラメーター --perform-sca-scan を使用してセンサーを再起動し、SCA スキャンを実行します。
- SCA スキャン・メタデータを保管するためのホスト上の追加ストレージ。センサースキャンのストレージ要件を参照してください。

仕組み

SCA スキャンは、デフォルトでは実行されません。ユーザーは、パラメーター --perform-sca-scan を使用して SCA スキャンを有効にする必要があります。有効にすると、コンテナイメージの標準脆弱性スキャン(静的または動的)の後に SCA スキャンが実行されます。SCA スキャンが完了すると、センサーはスキャンによって収集されたメタデータ情報を、ポスチャ評価が実行される Qualys バックエンドにアップロードします。SCA スキャン・データの検出結果は、イメージの詳細の一部として、Container Security UI および API で表示できます。SCA スキャンによって検出された脆弱性検出は、QID として表示されます。特定の脆弱性の検出に使用されるスキャンのタイプ (SCA、動的、または静的) を識別できるように、フィルターが用意されています。

インターネット・アクセスは SCA スキャンに対して使用可能であり、SCA スキャンはデフォルトでオンライン・モードで実行されます。センサーが URL「http://search.maven.org/」に到達できることを確認してください。

SCA スキャンでは、言語固有のソフトウェア・パッケージについて以下のファイルがスキャンされます。

Language	Files
Python	egg package wheel package
Node.js	package.json
.NET	packages.lock.json packages.config *.deps.json
Java	JAR/WAR/PAR/EAR
Go	Binaries built by Go
PHP	Composer.lock
Ruby	gemspec
Rust	Cargo.lock and Binaries built with cargo-auditable

シークレット検出

コンテナシークレットは、ID 認証を提供し、特権アカウント、アプリケーション、サービスへのアクセスを承認するデジタル認証情報です。これには、アプリケーションが正しく機能するために必要なパスワード、API キー、およびその他の資格情報を含めることができます。

これらのシークレットが適切に保護されていないと、権限のないユーザーがアクセスし、悪意のある攻撃につながる可能性があります。したがって、シークレットの検出は、機密データを保護し、コンプライアンス要件を満たし、セキュリティインシデントのリスクを軽減するために組織が優先する必要があるコンテナセキュリティの重要な側面の 1 つです。

Container Security Sensor は、コンテナイメージのシークレットを検出できるため、コンテナ内のシークレットの偶発的または意図的な公開に関連する潜在的なセキュリティリスクを軽減できます。

シークレット検出を有効にするには、**`--perform-secret-detection`** パラメーターを使用する必要があります。

シークレットの検出には、ファイルシステムのスキャンが含まれます。環境変数として格納されているシークレットや、イメージ内で引数として渡されたシークレットは検出されません。したがって、シークレット検出のパフォーマンスは、イメージに存在するファイルの数によって異なります。

シークレット検出で最適なパフォーマンスを得るには、センサー コンテナに割り当てる CPU 数を増やすことをお勧めします。ホストの少なくとも 2 つの CPU がセンサー コンテナに特別に使用されていることを確認します。

例えば：

- **InstallSensor.sh** スクリプトを使用する場合、既定では、ホストの CPU の 20% のみがセンサー コンテナによって使用されます。
- **dockerrun** を使用する場合、既定では、ホストのすべての CPU がセンサー コンテナに完全に使用されます

注: シークレット検出は、以下でのみサポートされます。

- センサー:CICD およびレジストリ
- OS:リナックス
- ランタイム: Docker、Containerd、および CRI-O

シークレット検出の詳細については、オンラインヘルプ: コンテナシークレットの検出を参照してください。

マルウェア検出

コンテナイメージをスキャンして、マルウェアや悪意のあるファイルの存在を検出できます。

マルウェア検出により、悪意のあるコンテナイメージが本番環境にデプロイされないようにします。これにより、悪意のあるコンテナの実行に起因する可能性のある潜在的な侵害、データの盗難、または不正アクセスが防止されます。

マルウェア検出を有効にするには、 センサーのインストール時に **--perform-malware-detection** パラメーターを使用する必要があります。

マルウェア スキャンを効率的に行うには、センサーに 1 つの CPU コアを割り当てることをお勧めします。例えば：

- **InstallSensor.sh** スクリプトを使用する場合、既定では、ホストの CPU の 20% がセンサー コンテナによって使用されます。ホストに 8 つの CPU コアがある場合、センサー コンテナに適用される合計 CPU 制限は 0.2 * 8 = 1.6 CPU コアになります。
- **dockerrun** を使用する場合、既定では、ホストのすべての CPU がセンサー コンテナに完全に使用されます。
- Kubernetes では、ホスト システムで使用可能なコアの数に関係なく、センサー コンテナに 1 つの CPU コアを割り当てるには、CPU 制限値を 1 に設定します。

```
Example:
resources:
limits:
cpu: "1"
```

注: マルウェア検出は、次の場合にのみサポートされます。

- センサー: レジストリ センサー (x86_64 アーキテクチャのみ)
- OS:リナックス
- ランタイム: Docker、Containerd、および CRI-O

マルウェア検出の詳細については、オンラインヘルプ : マルウェア検出を参照してください。

Docker アセットのスキャンにつながるイベント

新しいアセットスキャンは、次のいずれかのイベントが発生すると開始されます。

- イメージのイベント: load、pull、import、tag
- コンテナのイベント: start、create、unpause
- 新しいマニフェストがあると、スキャンが起動されます
- スキャンは、コンテナで 48 時間ごとに開始されます (つまり、最後に成功したスキャンから 48 時間後)

センサースキャンのストレージ要件

さまざまなスキャンタイプのストレージ要件を理解するには、以下のセクションを参照してください。

ダイナミックスキャン

レジストリ センサーにのみ適用されます。

レジストリー・センサーは、スキャンのためにホスト上の Docker イメージをプルします。Docker がインストールされているパーティションに必要なストレージは、イメージのサイズに基づきます。動的スキャンは、キャッシュされたイメージに対して実行されます。

平均的な画像サイズが 4 GB の場合、最大ストレージ要件は **16 GB** になります。

`4GB image * 4 scan threads = 16GB`

静的スキャン

汎用 (ホスト) センサー、レジストリ センサー、およびビルド (CI/CD) センサーに適用されます。

イメージにシェルがない場合、イメージをスキャンするには、永続ストレージに追加のストレージが必要です。静的スキャンは、コンテナ イメージに対して実行されます。ストレージ要件は、イメージのサイズの約 3 倍です。

平均イメージ サイズが 4 GB で、4 つのスキャン スレッドがシェルのないイメージに対してイメージ スキャンを実行している場合、最大ストレージ要件は **48 GB** になります。

`(4GB image * 3) * 4 scan threads = 48GB`

SCA スキャン

汎用 (ホスト) センサー、レジストリ センサー、およびビルド (CI/CD) センサーに適用されます。

CS センサーが **--perform-sca-scan** で実行されている場合、イメージ tar (通常はイメージのサイズに SCA スキャン メタデータの保存に使用される 100 MB の追加ディスク容量を加えたもの) を格納するために、ホストに追加のストレージが必要になります。必要なストレージは、イメージ サイズに Docker イメージ スキャンを実行するスレッド数の 100 MB を掛けた値です。

平均イメージ サイズが 4 GB で、4 つのスキャン スレッドがイメージ スキャンを実行している場合、最大ストレージ要件は約 **16.4 GB** です。

`(4GB image + 100MB) * 4 scan threads = 16.4GB`

静的 Log4j 検出

シェルを持つイメージに対して静的検出がトリガーされる場合は、追加のスペースが必要です。これには、画像の 3 倍のサイズが必要です。

平均的な画像サイズが 4 GB の場合、追加のストレージ要件は **12 GB** になります。

```
4GB image * 3 = 12GB
```

MacOS へのセンサーのインストール

注:MacOS Catalina バージョン 10.15 以降でセンサーを実行している場合は、センサーバージョン 1.8.0 以降を使用してください。

Qualys Container Sensor は MacOS にインストールできます。

QualysContainerSensor.tar.xz ファイルを使用して、ホームページの [コンテナ センサーのダウンロード] ボタンを使用するか、Qualys Cloud Platform の [構成] > [センサー] タブからファイルを作成します。

ターゲット MAC ホストにファイルをコピーします。次に、次のコマンドを順番に実行します。

次のコマンドは、tar ファイルを抽出します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

このコマンドは、構成、マニフェスト、ログ、セットアップなどのセンサー データが格納されているディレクトリを作成します。

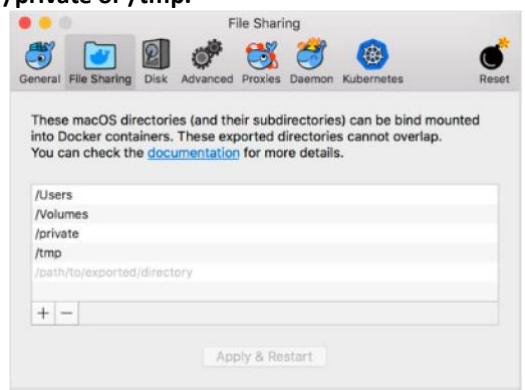
```
sudo mkdir -p /tmp/qualys/sensor/data
```

このコマンドは、インストーラースクリプトを実行するために必要なアクセス許可をディレクトリに提供します。

```
sudo chmod -R 777 /tmp/qualys/sensor/data
```

ストレージのカスタムの場所を指定する場合は、Docker のファイル共有が有効になっていることを確認してください。MAC ホストで、Docker > Preferences>File Sharing に移動し、カスタムパスを追加します。 **/usr/local/qualys/sensor/data, then click Apply & Restart.**

ファイル共有の有効化は、カスタムの場所が [NOT from] の場合にのみ必要です **/Users, /Volumes, /private or /tmp.**



この手順を回避するには、**Storage=/tmp/qualys/sensor/data** と **HostIdSearchDir=/private/etc/qualys** を使用します。これにより、CS センサーを起動するための追加の構成を必要とせずに、Docker

で既存の共有場所を活用できます。カスタムの場所を使用している場合は、インストーラー スクリプトを実行するためのアクセス許可をディレクトリに付与します。例えば：

```
sudo chmod -R 777 /usr/local/qualys/sensor/data
```

CS Sensor に/private/etc/qualys/ディレクトリの hostid ファイルへの書き込み権限があることを確認します。十分なアクセス許可を付与するには、次のコマンドを実行します。

```
sudo mkdir /private/etc/qualys/
sudo chmod 777 /private/etc/qualys
```

次のコマンドは、センサーをインストールします。コマンドには アクティベーション ID とお客様 ID は、どちらもサブスクリプションに基づいて生成されます。Storage パラメーターは、センサーの設置場所を指定します。HostIdSearchDir が存在することを確認しないと、インストーラースクリプトがエラーをスローします。

次のコマンドを使用して、汎用センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s
```

次のコマンドを使用して、レジストリ センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s
-registry-sensor
```

次のコマンドを使用して、CI/CD センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s
-
-cicd-deployed-sensor
```

注意：

- 永続ストレージなしでセンサーをインストールする場合は、「Storage」オプションを除外し、インストーラースクリプトに「**--sensor-without-persistent-storage**」オプションを含めます。データはホスト上では使用できませんが、センサーに関連する /usr/local/qualys/qpa/data フォルダーに保存されるため、「**--enable-console-logs**」オプションと「**--sensor-withoutpersistent-storage**」を使用してログを保持することをお勧めします。
- シークレットとマルウェアの検出は、MacOS ではサポートされていません。
-

Linux へのセンサーのインストール

Qualys Container Sensor は Linux にインストールできます。

QualysContainerSensor.tar.xz ファイルを使用して、ホームページの [コンテナ センサーのダウンロード] ボタンを使用するか、Qualys Cloud Platform の [構成] > [センサー] タブからファイルを作成します。

ファイルをターゲット・ホストにコピーします。次に、次のコマンドを順番に実行します。

次のコマンドは、tar ファイルを抽出します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

このコマンドは、構成、マニフェスト、ログ、セットアップなどのセンサー データが格納されているディレクトリを作成します。

```
sudo mkdir -p /usr/local/qualys/sensor/data
```

このコマンドは、必要な権限をストレージに追加します。

```
sudo chmod 777 /usr/local/qualys/sensor/data
```

次のコマンドは、センサーをインストールします。このコマンドには、サブスクリプションに基づいて生成されたアクティベーション ID とカスタマー ID が含まれていることに注意してください。Storage パラメーターは、センサーの設置場所を指定します。

次のコマンドを使用して、汎用センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/usr/local/qualys/sensor/data -s
```

次のコマンドを使用して、レジストリ センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/usr/local/qualys/sensor/data -s -r
```

次のコマンドを使用して、CI/CD センサーをインストールします。

```
sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/usr/local/qualys/sensor/data -s -c
```

注: 永続ストレージなしで Sensor をインストールするには、「Storage」オプションを除外し、インストーラー スクリプトに「**--sensor-without-persistent-storage**」オプションを含めます。データはホスト上では使用できませんが、センサーに関連する /usr/local/qualys/qpa/data フォルダーに

保存されるため、「**--enable-console-logs**」オプションと「**--sensor-withoutpersistent-storage**」を使用してログを保持することをお勧めします。

CoreOS へのセンサーのインストール

Qualys コンテナ センサーは CoreOS にインストールできます。ダウンロード

QualysContainerSensor.tar.xz ファイルを使用して、ホームページの [コンテナ センサーのダウンロード] ボタンを使用するか、Qualys Cloud Platform の [構成] > [センサー] タブからファイルを作成します。

ファイルをターゲット・ホストにコピーします。次に、次のコマンドを順番に実行します。

次のコマンドは、tar ファイルを抽出します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

このコマンドは、構成、マニフェスト、ログ、セットアップなどのセンサー データが格納されているディレクトリを作成します。

```
sudo mkdir -p /var/opt/qualys/sensor/data
```

注: ディレクトリ パス /var/opt/qualys/sensor/data を CoreOS で書き込み可能な Storage に設定する必要があります。

このコマンドは、インストーラースクリプトを実行するために必要なアクセス許可をディレクトリに提供します。

```
sudo chmod -R 777 /var/opt/qualys/sensor/data
```

次のコマンドは、センサーをインストールします。このコマンドには、サブスクリプションに基づいて生成されたアクティベーション ID とカスタマー ID が含まれていることに注意してください。Storage パラメーターは、センサーの設置場所を指定します。

次のコマンドを使用して、汎用センサーをインストールします。

```
Sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/var/opt/qualys/sensor/data/ -s
```

次のコマンドを使用して、レジストリ センサーをインストールします。

```
Sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/var/opt/qualys/sensor/data/ -s --registry-sensor
```

次のコマンドを使用して、CI/CD センサーをインストールします。

```
Sudo ./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-
8969e03cc58a4ef5 CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
Storage=/var/opt/qualys/sensor/data/ -s --cicd-deployed-sensor
```

注: 永続ストレージなしで Sensor をインストールするには、「Storage」オプションを除外し、インストーラー スクリプトに「**--sensor-without-persistent-storage**」オプションを含めます。データはホスト上では使用できませんが、センサーに関連する /usr/local/qualys/qpa/data フォルダーに保存されるため、「**--enable-console-logs**」オプションと「**--sensor-withoutpersistent-storage**」を使用してログを保持することをお勧めします。

Docker Hub を介したセンサーのインストール

このセクションでは、Docker Hub を介したセンサーのデプロイについて説明します。Docker Hub センサー イメージは、プライベート クラウド プラットフォーム (PCP) のお客様に対してはサポートされていないことに注意してください。

スタンドアロンの Docker ホストの場合:

docker compose を使用してスタンドアロンの Docker ホストにセンサーをデプロイする

docker run を使用してスタンドアロンの Docker ホストにセンサーをデプイする

Kubernetes の場合:

Docker Hub on Kubernetes を使用したセンサーのデプロイ

Docker Hub の Container Security Sensor は、次のように使用できます。

Qualys/QCS センサー:<tag>

Qualys/QCS センサー:latest

Docker Hub で最新のタグを検索します。

docker compose を使用してスタンドアロンの Docker ホストにセンサーをデプロイする

前提 条件：

- Docker エンジンバージョン:1.13.0+
- Docker-compose ファイル形式のバージョン:2.2
- Docker ホストは Docker Hub と通信できる必要があります

次の情報を含む新しい yml ファイルを作成します。ファイル名は **qualys_cs_sensor_docker_compose.yml** にできます。

注: yml ファイルのフィールド配置は非常に重要です。以下のテンプレートで提供されているフォーマットを必ず尊重してください。

```
version: '2.2'
services:
   cs_sensor:
       container_name: qualys-container-sensor
       image: qualys/qcs-sensor:latest
       restart: on-failure

# Uncomment the below security option if SELinux is enabled with
enforcing mode on docker host

#    security_opt:
```

```
#      - label:disable

# Enable the flag if you want to launch CS sensor in read-only mode.

#    read_only: true
   network_mode: host
   cpus: 0.2
   command: ["--scan-thread-pool-size", "4"]
   environment:
    - ACTIVATIONID=<Activation id>
    - CUSTOMERID=<Customer id>
    - POD_URL=<POD URL>

# Define TCP socket if sensor will be communicating with docker daemon
listening on TCP port
#  - DOCKER_HOST=<IPv4 address or FQDN>:<port#>
# Enable TLS authentication if sensor will be communicating with docker
daemon over TCP TLS socket
#      - DOCKER_TLS_VERIFY=1
# Define the proxy if required
#  -  qualys_https_proxy=<IP address or FQDN>:<Port#>

     volumes:

# Provide host Id search directory path

    -  /etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data

# Mount volume for docker socket

   - /var/run/docker.sock:/var/run/docker.sock:ro

# Mount volume for persistent storage

   - /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data

# Mount volume proxy certificate if required

#  - <Proxy certificate path on host>:/etc/qualys/qpa/cert/customca.crt

# Mount volume for docker client cert directory path

#  - <Client certificate directory on the docker host>:/root/.docker
```

yml ファイルで使用されるパラメーター

container_name

set to qualys-container-sensor

image_name

set to qualys/qcs-sensor:<tag>

OR

set to qualys/qcs-sensor:latest

イメージは、docker-compose によって Docker Hub からプルされます。

restart

センサーの再起動ポリシーを定義し、on-failure に設定する必要があります。

security_opt

このパラメータは、Docker ホストで SELinux が強制モードで有効になっている場合にのみ使用してください。

```
security_opt:      -
label:disable
```

readonly

センサーを読み取り専用モードで起動する場合は true に設定します。

network_mode

ホストのネットワーク スタックでセンサーを起動する必要があることを指定する host に設定します。

cpus

CPU 使用率を特定の値に制限します。

cpus: 0.2 # デフォルトの CPU 使用率制限 (ホスト上の 1 つのコア/プロセッサの 20%)。

たとえば、CPU 使用率を 5% に制限するには、cpus: 0.05 を設定します。これにより、CPU 使用率がホスト上の 1 つのコア/プロセッサの 5% に制限されます。

ノードに複数のプロセッサがある場合、cpus 値を設定すると、CPU 制限が 1 つのコア/プロセッサのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、cpus 値を 0 に設定するか、削除/コメントアウトします。

注: Docker ホストのカーネルが実行中のコンテナの CPU 制限の設定をサポートしていない場合は、CPU 使用率の制限を無効にしないと、センサーが起動されません。

Command

CI/CD 環境用にセンサーをデプロイする場合は、コマンド値を次のように指定します。

```
command: ["--cicd-deployed-sensor"]
```

Registry Sensor をデプロイする場合は、コマンド値を次のように指定します。

```
command: ["--registry-sensor"]
```

注: レジストリまたは CI/CD のパラメーターが指定されていない場合、General Sensor はデフォルトでインストールされます。

コマンド・パラメーターで指定できる追加の値:

"--enable-console-logs" を使用して、コンソールにログを出力します。これらのログは、docker logs コマンドを使用して取得できます。

"--log-level" は、センサーのログ レベルを設定し、0 から 5 を受け入れます。既定値は 3 (情報) です。

"--log-filesize" は、センサーのログファイルあたりの最大サイズをバイト単位で設定します。受け入れる

"<digit><K/M/>" ここで、K はキロバイト、M はメガバイトです。たとえば、10 バイトの場合は「10」、10 キロバイトの場合は「10K」、10 メガバイトの場合は「10M」を指定します。デフォルトは「10M」です。"--log-filepurgecount" は、生成されるアーカイブされた qpa.log ファイルの数を定義します。既定値は 5 です。

"--scan-thread-pool-size" を使用して、スキャン スレッド値でセンサーを起動します。既定値は 4 です。

"--sensor-without-persistent-storage" は、ホスト上の永続ストレージを使用せずにセンサーを実行します。この場合、ボリュームの下に永続ストレージマッピングを提供しないでください。データはホスト上では使用できませんが、センサーに関連する /usr/local/qualys/qpa/data フォルダーに保存されるため、「--enable-console-logs」オプションと「--sensor-withoutpersistent-storage」を使用してログを保持することをお勧めします。例

```
command: ["--cicd-deployed-sensor", "--sensor-without-persistent-
storage", "--enable-console-logs"]
volumes:
            # mount volume for persistent storage
            # -/usr/local/qualys/qpa/data
```

"--tls-cacert","<file name of the CA certificate used to sign docker server certificate>", "-tls-cert", "<docker client certificate file name>", "--tls-key", "<docker client private key file name>" (センサーが TLS 経由で Docker デーモンと通信する場合)。3 つのファイルのいずれかにそれぞれ ca.pem、cert.pem、key.pem などのデフォルト名がある場合は、対応する引数を省略できます。

"--mask-env-variable" を使用して、イメージとコンテナの環境変数をマスクします。環境変数は、センサー ログと Container Security UI でマスク/削除されます。

"--disableImageScan" は、General Sensor のイメージ スキャンを無効にします。このオプションで展開されたセンサーでは、画像はスキャンされません。これは、一般的なセンサータイプでのみ使用でき、すべてのランタイム(Docker、CRI-O、Containerd)で使用できます。

"--disable-log4j-scanning" は、コンテナイメージの Log4j 脆弱性スキャンを無効にします。Log4j 脆弱性スキャンを参照してください。

"--disable-log4j-static-detection" は、動的/静的イメージ スキャンの log4j 静的検出を無効にします。静的 log4j 検出を参照してください。

"--optimize-image-scans" は、General Sensor のイメージスキャンを最適化します。デフォルトでは、センサーはホストで検出したすべての画像をスキャンします。これにより、イメージのスキャンが冗長になります。「--optimize-image-scans」を使用して一般センサーをインストールすると、センサーは Qualys Cloud Platform と通信し、情報に基づいたスキャンを実行して、冗長なイメージ スキャンを回避します。センサーは、ホストに存在するイメージが、同じマニフェストとバージョンについて他のセンサーによって既にスキャンされているかどうかを判断し、それらのイメージを再度スキャンしません。

"--perform-secret-detection" を使用して、コンテナ イメージのシークレット検出を有効にします。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

"--perform-malware-detection" を使用して、コンテナ イメージのマルウェア検出を有効にします。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

environment

サブスクリプションの ACTIVATIONID、CUSTOMERID、POD_URL を指定します。アクティベーション ID とカスタマー ID を取得するには、コンテナ セキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

注: ホストは、HTTPS ポート 443 を介して Qualys Cloud Platform (または Qualys Private Cloud Platform) に到達できる必要があります。ホストがアクセスする必要がある Qualys プラットフォーム(POD URL)を参照してください。

センサーが TCP ポートでリッスンしている Docker デーモンと通信するかどうか DOCKER_HOST、TLS の有無にかかわらず指定します。

```
DOCKER_HOST=<IPv4 address, or FQDN, or hostname>:<Port#>
```

指定された TCP ソケットに対して TLS が有効になっている場合は、指定された IP、FQDN、またはホスト名が、Docker サーバー証明書の CN または代替サブジェクト名と一致することを確認してください。

センサーが TLS なしで TCP ソケットでリッスンしている場合は、UNIX ドメインソケットディレクトリマッピングを提供しません。「ボリューム」の下で、次の部分をコメントアウトします。

```
volumes:
# mount volume for docker socket
#  - /var/run/docker.sock:/var/run/docker.sock:ro
```

DOCKER_TLS_VERIFY=1 を指定して、TLS 認証を有効にします。

注: TLS 経由で Docker デーモンとのセンサー通信を有効にすることで、お客様は docker 認証プラグインを使用して docker ソケットへのセンサーのアクセスを制限できます。

センサーが Qualys Cloud Platform と通信するためにプロキシが必要かどうかを指定します。
qualys_https_proxy

```
- qualys_https_proxy=<IP/ address or FQDN>:<Port#>
```

volumes

永続ストレージ マッピングを指定して、永続ストレージでセンサーを起動します。永続ストレージディレクトリが存在しない場合は、自動的に作成されます。

```
volumes:
# mount volume for persistent storage
  -/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data
```

以前のインストールで使用していたものと同じホスト ID を使用する場合は、hostid ディレクトリの場所を指定します。

```
# Provide host Id search directory path
- /etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data
```

Docker ホスト上の Docker デーモンが Unix ソケット経由で通信している場合は、Unix ソケット ファイルをセンサー ファイル システムにマップします。

```
# mount volume for docker socket
- /var/run/docker.sock:/var/run/docker.sock:ro
```

注: Docker デーモンが TCP ポート経由で通信している場合は、environment で DOCKER_HOST パラメーターを指定し、volumes で docker unix ソケット ファイルのマッピングを指定しないでください。

プロキシ証明書を指定します (必要な場合)。

```
- <Proxy certificate path on host>:/etc/qualys/qpa/cert/custom-ca.crt
```

センサーが TLS 経由で Docker デーモンと通信する場合は、Docker クライアント証明書ディレクトリ マッピングを指定します。

```
# mount volume for docker client certificate directory
- <docker client certificate directory on the docker daemon
  host>:/root/.docker
```

センサーの起動

yml ファイルが作成されたら、次のコマンドを使用してセンサーを起動します。

```
docker-compose -f <path to qualys_cs_sensor_docker_compose.yml file> up
-d
```

センサーのアップグレード

Docker Hub でホストされている Qualys Container Sensor イメージは、自動更新をサポートしていません。Docker Hub からインストールされたセンサーを更新するには、次の手順を実行します。

1. yml ファイルのイメージ名を更新します。

   qualys/qcs-sensor:<tag >

   又は
   qualys/qcs-sensor:latest に設定

2. 次のコマンドを実行して、センサーを再作成します。

```
docker-compose -f <path to qualys_cs_sensor_docker_compose.yml file>
up -d
```

センサーの取り外し

次のコマンドを実行して、センサーを取り外します。

```
docker-compose -f <path to qualys_cs_sensor_docker_compose.yml file>
rm -s
```

注: docker-compose には、永続ストレージを削除するオプションはありません。永続ストレージファイルを手動で削除する必要があります。

docker run を使用してスタンドアロンの Docker ホストにセンサーをデプロイする

前提条件: Docker エンジンのバージョン: 1.13.0+

次のコマンドを実行して、センサーをインストールします。ACTIVATIONID を指定します。

CUSTOMERID、およびサブスクリプションから POD_URL されます。アクティベーション ID を取得し、カスタマー ID で Container Security UI にログインし、[Configurations > Sensors] に移動して、センサーをダウンロードします。センサーの種類 (全般、レジストリ、CI/CD) を選択し、スタンドアロンテクノロジ (MacOS、Linux、または CoreOS) を選択します。[Installation Instructions](インストール手順)ページが表示されます。[DOCKERHUB] タブを選択して、インストール手順を確認します。

[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。注: CIS ベンチマーク 5.9 に準拠するには、インストール コマンドから --net=host を削除する必要があります。ただし、センサーを --net=host なしで起動すると、ホストの IP アドレスを検出できないことに注意してください。 ガイダンスと推奨事項については、「Docker の CIS ベンチマークへの準拠」を参照してください。

General Sensor

Linux:

```
sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD URL> --net=host --name qualys-container-sensor qualys/qcs-
sensor:latest
```

MacOS:

```
sudo mkdir -p /tmp/qualys/sensor/data
sudo chmod -R 777 /tmp/qualys/sensor/data
sudo mkdir /private/etc/qualys/ sudo
chmod 777 /private/etc/qualys

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/private/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data -v
/tmp/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD URL> --net=host --name qualys-container-sensor qualys/qcs-
sensor:latest
```

CoreOS:

```
sudo mkdir -p /var/opt/qualys/sensor/data
sudo chmod -R 777 /var/opt/qualys/sensor/data

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/var/opt/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD URL> --net=host --name qualys-container-sensor qualys/qcs-
sensor:latest
```

Registry Sensor

Linux:

```
sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--registry-sensor
```

MacOS:

```
sudo mkdir -p /tmp/qualys/sensor/data
sudo chmod -R 777 /tmp/qualys/sensor/data
sudo mkdir /private/etc/qualys/ sudo
chmod 777 /private/etc/qualys

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/private/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data -v
/tmp/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--registry-sensor
```

CoreOS:

```
sudo mkdir -p /var/opt/qualys/sensor/data
sudo chmod -R 777 /var/opt/qualys/sensor/data

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/var/opt/qualys/sensor/data:/usr/local/qualys/qpa/data -e
```

```
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--registry-sensor
```

CI/CD Sensor

Linux:

```
sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--cicd-deployed-sensor
```

MacOS:

```
sudo mkdir -p /tmp/qualys/sensor/data
sudo chmod -R 777 /tmp/qualys/sensor/data
sudo mkdir /private/etc/qualys/ sudo
chmod 777 /private/etc/qualys

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/private/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data -v
/tmp/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--cicd-deployed-sensor
```

CoreOS:

```
sudo mkdir -p /var/opt/qualys/sensor/data
sudo chmod -R 777 /var/opt/qualys/sensor/data

sudo docker run -d --restart on-failure -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/var/opt/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e
POD_URL=<POD
URL> --net=host --name qualys-container-sensor qualys/qcs-sensor:latest
--cicd-deployed-sensor
```

上記のコマンドで使用されるボリューム:

/var/run/docker.sock:/var/run/docker.sock:ro - Docker ソケットをセンサー ファイル システムにマウントします。これは、docker デーモンが TCP ポートで実行されている場合に、ユーザーが DOCKER_HOST 環境変数を指定しない限り必須です。

/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data - センサー コンテナに永続ストレージを提供します。このマッピングは、「--sensor-without-persistent-storage」オプションが使用されていない限り必須です。ストレージディレクトリは変更できます。ディレクトリが存在しない場合は、自動的に作成されます。

追加の環境変数/ボリュームを指定できます。

1) プロキシを使用して Qualys Cloud Platform と通信する場合は、以下を指定します。

```
-e qualys_https_proxy=<IP/ address or FQDN>:<Port#>
```

2)プロキシ証明書が必要な場合は、以下を追加してプロキシ証明書のボリュームをマウントします。

```
-v <Proxy_File_Path>:/etc/qualys/qpa/cert/custom-ca.crt
```

3)Docker デーモンが TCP ポートで実行されている場合は、-e DOCKER_HOST=<IPv4 アドレスまたは FQDN>:<port#> を指定します。この場合、Docker Unix ドメイン ソケット ボリューム マウント (**-v /var/run/docker.sock:/var/run/docker.sock:ro**) を必ず削除してください。

/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data - HostID search directory to map the marker file created by Qualys Agent or Scanner appliance on the host.

4)センサーが TLS ソケット経由で Docker デーモンと通信する場合は、次の必須環境変数とボリューム マウントを指定します。

環境変数を設定して、接続先の TLS Docker ソケット DOCKER_HOST 指定します。

```
-e DOCKER_HOST=<docker daemon host's IPv4  address, or FQDN, or
hostname>:<port#>
```

指定された IPv4 アドレス、FQDN、またはホスト名が、Docker サーバー証明書の CN または代替サブジェクト名と一致する場合。

TLS 認証セットを有効にするには:

```
-e DOCKER_TLS_VERIFY=1
```

注: TLS 経由で Docker デーモンとのセンサー通信を有効にすることで、お客様は docker 認証プラグインを使用して docker ソケットへのセンサーのアクセスを制限できます。

ボリュームマウント Docker デーモンホスト上のディレクトリには、Docker クライアント証明書、クライアント秘密鍵、および CA 証明書ファイルが使用可能です。

```
-v <docker client certificate directory on the docker daemon
host>:/root/.docker
```

センサーの引数として、Docker クライアント証明書、クライアント秘密鍵、および CA 証明書のファイル名を指定します。

```
--tls-cacert <file name of the CA certificate used to sign docker server certificate> --tls-cert <docker client certificate file name> --tls-key <docker client private key file name>
```

CA 証明書、クライアント証明書、またはクライアント秘密キーのいずれかに、それぞれ ca.pem、cert.pem、key.pem などのデフォルトのファイル名がある場合は、省略できます。たとえば、docker daemon が unix ドメイン ソケットと TCP TLS ソケットの両方でリッスンしている場合は、次のようにセンサーを起動できます。

```
docker run -d --restart on-failure --cpus=0.2 -v

/var/run/docker.sock:/var/run/docker.sock:ro -v <client cert directory on the docker host>:/root/.docker -v

/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e

ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> -e DOCKER_TLS_VERIFY=1 -e DOCKER_HOST=<IPv4 or FQDN>:<port#> -net=host --name qualys-container-sensor qualys/qcs-sensor:latest --loglevel 5 --tls-cacert <file name of the CA certificate used to sign docker server certificate> --tls-cert <docker client certificate file name> -tls-key <docker client private key file name>
```

Optional Parameters

--cpus

CPU 使用率を特定の値に制限します。

--cpus=0.2 # デフォルトの CPU 使用率制限 (ホスト上の 1 つのコア/プロセッサの 20%)。

たとえば、CPU 使用率を 5% に制限するには、--cpus=0.05 を設定します。これにより、CPU 使用率がホスト上の 1 つのコア/プロセッサの 5% に制限されます。

ノードに複数のプロセッサがある場合、cpus 値を設定すると、CPU 制限が 1 つのコア/プロセッサのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、オプションを指定しないでください。

注: Docker ホストのカーネルが実行中のコンテナの CPU 制限の設定をサポートしていない場合は、CPU 使用率の制限を無効にしないと、センサーが起動されません。

--enable-console-logs

コンソールにログを出力します。これらのログは、docker logs コマンドを使用して取得できます。

--sensor-without-persistent-storage

ホスト上の永続ストレージを使用せずにセンサーを実行します。この場合、ボリュームの下に永続ストレージマッピングを提供しないでください。データはホスト上では使用できませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されるため、「--enableconsole-logs」オプションと「--sensor-without-persistent-storage」を使用してログを保持することをお勧めします。

--log-level

センサーのロギングレベルを設定し、0〜5 を受け入れます。既定値は 3 (情報) です。

--log-filesize

センサーのログ ファイルあたりの最大サイズをバイト単位で設定します。たとえば、10 バイトの場合は「10」、10 キロバイトの場合は「10K」、10 メガバイトの場合は「10M」を指定します。デフォルトは「10M」です。

--log-filepurgecount

生成するアーカイブされた qpa.log ファイルの数を定義します。既定値は 5 です。

--scan-thread-pool-size

スキャンスレッド値でセンサーを起動します。既定値は 4 です。

--read-only

センサーを読み取り専用モードで実行します。このモードでは、センサーはホスト上の永続ストレージを使用します。

注: センサーは、「--sensor-without-persistent-storage」オプションまたは「--read-only」オプションのいずれかを使用して実行し、両方のオプションを同時に有効にしないでください。

--mask-env-variable

イメージとコンテナの環境変数をマスクします。環境変数は、センサー ログと Container Security UI でマスク/削除されます。

--disableImageScan

このパラメータは、General Sensor のイメージスキャンを無効にする場合に渡す必要があります。

このオプションで展開されたセンサーでは、画像はスキャンされません。これは、一般的なセンサータイプでのみ使用でき、すべてのランタイム(Docker、CRI-O、Containerd)で使用できます。

--disable-log4j-scanning

このパラメータは、コンテナイメージの log4j 脆弱性スキャンを無効にする場合に渡す必要があります。「Log4j 脆弱性スキャン」を参照してください。

--disable-log4j-static-detection

このパラメータは、動的/静的イメージスキャンの log4j 静的検出を無効にする場合に渡す必要があります。静的 log4j 検出を参照してください。

--optimize-image-scans

このパラメータは、General Sensor の画像スキャンを最適化する場合に渡す必要があります。デフォルトでは、センサーはホストで検出したすべての画像をスキャンします。これにより、画像のスキャンが冗長になります。「--optimizeimage-scans」を使用して一般センサーをインストールすると、センサーは Qualys クラウドプラットフォームと通信し、情報に基づいたスキャンを実行して、冗長なイメージスキャンを回避します。センサーは、ホストに存在するイメージが、同じマニフェストとバージョンについて他のセンサーによって既にスキャンされているかどうかを判断し、それらのイメージを再度スキャンしません。

--scanning-policy

このパラメータは、スキャンポリシーを指定する場合に渡す必要があります。スキャンポリシーでは、要件に応じて適切なスキャンタイプを選択できます。使用可能な値は次のとおりです。

- **DynamicScanningOnly**: 動的スキャンのみを実行します。
- **StaticScanningOnly:** 静的スキャンのみを実行します。
- **DynamicWithStaticScanningAsFallback**: シェルを使用しないイメージの動的スキャンのフォールバックとして静的スキャンを実行します。

--perform-secret-detection

このパラメーターは、コンテナ イメージのシークレット検出を実行する場合に渡す必要があります。シークレット検出のタイムアウトは、`--sca-scantimeout-in-seconds={value}` パラメーターを使用して指定できます。シークレットの検出の詳細については、オンラインヘルプ: コンテナシークレットの検出を参照してください。

--perform-malware-detection

このパラメーターは、コンテナ イメージのマルウェア検出を実行する場合に渡す必要があります。マルウェア検出のタイムアウトは、**`--sca-scantimeout-in-seconds={value}`** パラメーターを使用して指定できます。マルウェア検出の詳細については、オンラインヘルプ : マルウェア検出を参照してください。

SCA スキャンのオプション・パラメーター

以下のパラメーターは、サブスクリプションで SCA スキャン機能が有効になっている場合はオプションです。詳しくは、SCA スキャンを参照してください。

--perform-sca-scan

(オプション)デフォルトでは、SCA スキャンは実行されません。このパラメーターを使用して、コンテナ・イメージの SCA スキャンを使用可能にします。指定すると、SCA スキャンは標準の脆弱性スキャン (静的または動的) の後に実行されます。SCA スキャンは、脆弱性スキャンが成功しない場合でも試行されます。

--disallow-internet-access-for-sca

(--perform-sca-scan が指定されている場合はオプション)デフォルトでは、SCA スキャンはオンライン・モードで実行されます。このパラメーターを使用して、SCA スキャンのインターネット・アクセスを使用不可にし、スキャンをオフライン・モードで実行します。

注: SCA スキャンはオンライン・モードで実行することをお勧めします。Java のソフトウェア・パッケージ列挙の品質は、SCA スキャンがオフライン・モードで実行されると大幅に低下します。正確なパッケージ検出のために、リモートの Maven リポジトリを参照する必要がある場合があります。これは、イメージの脆弱性ポスチャの精度に影響を与える可能性があります。

--sca-scan-timeout-in-seconds={value}

(--perform-sca-scan が指定されている場合はオプション)デフォルトの SCA スキャン・コマンド・タイムアウトは 5 分 (300 秒) です。このパラメーターを使用して、デフォルトのタイムアウトを秒単位で指定した新しい値で上書きします。例えば、大きなコンテナ・イメージをスキャンするときには、SCA スキャンが完了する時間を確保するために、SCA スキャンのタイムアウトを増やす必要がある場合があります。

注: **`--sca-scan-timeout-in-seconds`** パラメータは、シークレットとマルウェア検出のタイムアウトを指定するためにも使用されます。

Docker 実行コマンドを使用して Docker の CIS ベンチマークに準拠する方法

Qualys Container Security は、センサー イメージの Docker の CIS ベンチマークに準拠しています。Docker の CIS ベンチマークに準拠する方法で Sensor イメージを使用する方法については、Docker の CIS ベンチマークへの準拠を参照してください。準拠した方法でセンサーを操作できるように、いくつかのコントロールの手順を提供しています。

Kubernetes 上の Docker Hub を使用したセンサーのデプロイ

前提 条件：

- Kubernetes のセットアップが稼働している状態である
- K8S ノードは、Docker ハブ/プライベートレジストリと通信できる状態である
- コンテナ センサー イメージは、プライベート レジストリからインストールする場合は、プライベート レジストリで使用する必要がある

cssensor-ds.yml ファイルを変更する

次の情報を含む新しい yml ファイルを作成し、**cssensor-ds.yml** という名前を付けるか、https://github.com/Qualys/cs_sensor から直接 yml ファイルをダウンロードします。

注: yml ファイルのフィールド配置は非常に重要です。以下のテンプレートで提供されているフォーマットを必ず使用してください。

```
kind: List
apiVersion: v1
items:
- kind: Namespace
apiVersion: v1
metadata:
name: qualys
# Service Account
- kind: ServiceAccount
apiVersion: v1
metadata:
name: qualys-service-account
namespace: qualys
# Role for read/write/delete permission to qualys namespace
- kind: Role
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-reader-role
namespace: qualys
rules:
- apiGroups: ["","batch"]
resources: ["pods","jobs"]
verbs: ["get", "list", "watch","create", "delete",
"deletecollection"]
- apiGroups: [""]
resources: ["pods/status"]
verbs: ["get"]
- apiGroups: [""]
resources: ["pods/attach", "pods/exec"]
verbs: ["create"]
- kind: ClusterRole
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-cluster-reader-role
rules:
- apiGroups: [""]

resources: ["nodes", "pods/status",
"replicationcontrollers/status", "nodes/status"]
```

```
verbs: ["get"]
- apiGroups: ["apps"]
resources: ["replicasets/status", "daemonsets/status",
"deployments/status", "statefulsets/status"]
verbs: ["get"]
- apiGroups: ["batch"]
resources: ["jobs/status", "cronjobs/status"]
verbs: ["get"]
# RoleBinding to assign permissions in qualys-reader-role to qualys-
service-account
- kind: RoleBinding
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-reader-role-rb
namespace: qualys
subjects:
- kind: ServiceAccount
name: qualys-service-account
namespace: qualys
roleRef:
kind: Role
name: qualys-reader-role
apiGroup: rbac.authorization.k8s.io
- kind: ClusterRoleBinding
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-cluster-reader-rb
subjects:
- kind: ServiceAccount
name: qualys-service-account
namespace: qualys
roleRef:
kind: ClusterRole
name: qualys-cluster-reader-role
apiGroup: rbac.authorization.k8s.io
# Qualys Container Sensor pod with
- apiVersion: apps/v1
kind: DaemonSet
metadata:
name: qualys-container-sensor

namespace: qualys
labels:
k8s-app: qualys-cs-sensor
spec:
selector:
matchLabels:
name: qualys-container-sensor
```

```
updateStrategy:
type: RollingUpdate
template:
metadata:
labels:
name: qualys-container-sensor
spec:
#tolerations:
# this toleration is to have the daemonset runnable on master
nodes
# remove it if want your masters to run sensor pod
#- key: node-role.kubernetes.io/master
# effect: NoSchedule
serviceAccountName: qualys-service-account
containers:
- name: qualys-container-sensor
image: qualys/qcs-sensor:latest
imagePullPolicy : IfNotPresent
resources:
limits:
cpu: "0.2" # Default CPU usage limit on each node for
sensor.
args: ["--k8s-mode"]
env:
- name: CUSTOMERID
value: __customerId
- name: ACTIVATIONID
value: __activationId
- name: POD_URL
value:
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
value: "10"
# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- name: DOCKER_TLS_VERIFY
# value: "1"
# uncomment(and indent properly) below section if proxy is required to
connect Qualys Cloud
#- name: qualys_https_proxy
# value: <proxy FQDN or Ip address>:<port#>
- name: QUALYS_POD_NAME
valueFrom:
fieldRef:
fieldPath: metadata.name
- name: QUALYS_POD_NAMESPACE
valueFrom:
fieldRef:
fieldPath: metadata.namespace
volumeMounts:
- mountPath: /var/run/docker.sock
name: socket-volume
```

```
readOnly: true
- mountPath: /usr/local/qualys/qpa/data
name: persistent-volume
- mountPath: /usr/local/qualys/qpa/data/conf/agent-data
name: agent-volume
# uncomment(and indent properly) below section if proxy(with CA cert)
required to connect Qualys Cloud
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
# name: proxy-cert-path
# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- mountPath: /root/.docker
# name: tls-cert-path
securityContext:
allowPrivilegeEscalation: false
volumes:
- name: socket-volume
hostPath:
path: /var/run/docker.sock
type: Socket
- name: persistent-volume
hostPath:
path: /usr/local/qualys/sensor/data
type: DirectoryOrCreate
- name: agent-volume
hostPath:
path: /etc/qualys
type: DirectoryOrCreate
# uncomment(and indent properly) below section if proxy(with CA cert)
required to connect Qualys Cloud
#- name: proxy-cert-path
# hostPath:
# path: <proxy certificate path>
# type: File

# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- name: tls-cert-path
# hostPath:
# path: <Path of directory of client certificates>
# type: Directory
hostNetwork: true
```

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切な権限を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、まず他の名前空間から Qualys Container Sensor をアンインストールしてから、「qualys」名前空間に新たにデプロイする必要があります。

次の権限が必要です: get、list、watch - 'qualys' 名前空間のリソースを監視するため create、delete、deleteCollection - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成し、その後にクリーンアップします。

yaml ファイルのパラメーターを変更する

cssensor-ds.yml ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください。

spec の tolerations セクションのコメントを解除して、Sensor daemonset をマスターノードにデプロイします。

```
spec:
#tolerations:
# this toleration is to have the daemonset runnable on master nodes
# remove it if want your masters to run sensor pod
#- key: node-role.kubernetes.io/master
# effect: NoSchedule
```

yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてください。すべての Kubernetes ノードから、Docker ハブ/プライベートレジストリ(CS Sensor イメージが公開されている場所)にアクセスできることを確認します。

```
containers:
      - name: qualys-container-sensor
        image: <CS Sensor image name in the private/docker hub registry>
    args: ["--k8s-mode"]
```

注: センサー ポッドを実行するすべてのノードが、センサー イメージが格納されているプライベート レジストリまたは Docker ハブ レジストリにアクセスできることを確認してください。

CI/CD 環境用にセンサーをデプロイする場合は、args 値を次のように指定します。

```
 args: ["--k8s-mode","--cicd-deployed-sensor"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
 ["--k8s-mode","--registry-sensor"]
```

コンソールでログを出力する場合は、args の追加値として "**--enable-console-logs**" を指定します。

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
args: ["--k8s-mode", "--log-level", "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
args: ["--k8s-mode", "--scan-thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数を定義する場合は、args の追加値として "**--logfilepurgecount**", "" を指定します。デフォルトの "**--logfilepurgecount**", "5" は config で適用されます。log/ ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。生成されるアーカイブされた qpa.log ファイルの数とログ ファイルごとのサイズを定義する場合は、args の追加値として "**--log-filesize**", "" ("K" はキロバイト、"M" はメガバイトを意味し、"**--log-filepurgecount**", "" を指定します。既定値は "**--log-filesize**": "10M" と "**--log-filepurgecount**": "5" で、config を介して適用されます。

```
args: ["--k8s-mode", "--log-filesize", "5M", "--logfilepurgecount",
"4"]
```

kubernetes ネイティブの 'kubectl run' コマンドを使用してイメージ スキャン ポッドをインスタンス化する場合は、args の追加値として "**--use-kubectl**" を指定します。この場合、センサーはネイティブの kubernetes 機能を使用してイメージ スキャンを開始します。この引数を省略すると、イメージ コンテナは docker run を使用して起動されます。

```
args: ["--k8s-mode", "--use-kubectl"]
```

TLS 認証が有効になっている場合は、引数に Docker クライアント証明書、クライアント秘密鍵、および CA 証明書の名前を指定します。

```
args: ["--k8s-mode", "--tls-cacert","<file name of the CA certificate
that was used to sign docker server certificate>", "--tls-cert",
"<docker client certificate file name>", "--tls-key", "<docker client
private key file name>"]
```

注: 3 つのファイルのいずれかにそれぞれ ca.pem、cert.pem、key.pem などのデフォルト名がある場合は、対応する引数を省略できます。

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「**--mask-env-variable**」パラメータを追加します。

```
args: ["--k8s-mode", "--mask-env-variable"]
```

General Sensor(すべてのランタイムでサポート)のイメージスキャンを無効にする場合は、args に「**--disableImageScan**」パラメーターを追加します。

```
args: ["--k8s-mode", "--disableImageScan"]
```

スキャンポリシーを指定する場合は、args に "**--scanning-policy**" パラメーターを追加します。スキャン ポリシーで使用できる値は、"DynamicScanningOnly"、"StaticScanningOnly"、および "DynamicWithStaticScanningAsFallback" です。

```
args: ["--k8s-mode", "--scanning-policy", "StaticScanningOnly"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「**--disable-log4j-scanning**」パラメータを追加します。

```
args: ["--k8s-mode", "--disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "**--disable-log4j-static-detection**" パラメーターを追加します。

```
args: ["--k8s-mode", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「**--optimize-image-scans**」パラメーターを追加します。

```
args: ["--k8s-mode", "--optimize-image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "**--perform-secretdetection**" パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
performsecret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "**--performmalware-detection**" パラメーターを追加します。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-malware-
detection"]
```

[リソース] で 、次の項目を指定します。

```
resources: limits: cpu: "0.2" # Default CPU usage limit on
each node for sensor.
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
    resources:limits:
        cpu: "0.2" # Default CPU usage limit on each node for sensor
```

```
  memory: "500Mi"
  requests:
  memory: "300Mi"
```

Container Sensor にメモリリソースの値(制限またはリクエスト)のいずれかが指定され、args に「--use-kubectl」が指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required)
- name: ACTIVATIONID
       value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required)
- name: CUSTOMERID
      value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL      value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
  value: "30"
```

TLS 認証を有効にするには、次の 2 行のコメントを解除します。

```
- name: DOCKER_TLS_VERIFY      value: "1"
```

注:TLS を無効にするには、DOCKER_TLS_VERIFY=""(空の文字列)を使用するか、yml ファイルでコメントを削除または保持します。

注: TLS 経由で Docker デーモンとのセンサー通信を有効にすることで、お客様は docker 認証プラグインを使用して docker ソケットへのセンサーのアクセスを制限できます。

注:TLS 認証が有効で、DOCKER_HOST が指定されていない場合、センサーは実行されているワーカー・ノードの FQDN を自動的に検出し DOCKER_HOST センサー内の環境変数を<ワーカー・ノードの FQDN>:<2376>に設定します(2376 は Docker デーモンのデフォルトの TLS TCP ポートです)

注: 以下を追加することで、DOCKER_HOST を 127.0.0.1:<port#> または localhost:<port#> に設定できます。

```
- name: DOCKER_HOST
      value: "<loopback IPv4 address or hostname>:<port#>"
```

注: DOCKER_HOST に設定されている FQDN、ホスト名、または IPv4 アドレスが、各ワーカー ノードの Docker サーバー証明書の CN またはサブジェクトの別名と一致していることを確認してください。

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

TLS 認証を有効にする必要がある場合は、ボリュームの下の tls-cert-path のコメントを解除し 、クライアント証明書のディレクトリパスを指定するか、不要な場合はそのままにしておきます。

```
#- name: tls-cert-path
# hostPath:
# path: <Path of directory of client certificates>
# type: Directory
```

ボリュームの下の proxy-cert-path のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
# hostPath:
# path: /root/cert/proxy-certificate.crt
# type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
# name: proxy-cert-path
```

TLS を使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /root/.docker
# name: tls-cert-path
```

Remove hostNetwork: IP 識別が不要な場合は true

hostNetwork: true **オプションを削除/コメントアウト** して、ホストの IP アドレスの識別が不要な最小特権のセキュリティのベスト プラクティスに従います。

```
hostNetwork: true
```

hostNetwork: true オプションは、Docker ホストの IP アドレスを検出する機能をセンサーに提供します。ユーザーは、yaml ファイルの hostNetwork:true 行を削除/コメントアウトできますが、欠点は、UI が Docker ブリッジネットワークの IP アドレスをホスト IP アドレスとして表示することです。ホストの識別を支援するために、ホスト名が UI に表示されるため、お客様はホスト名で Docker ホストを識別できます。また、ホスト名は環境内で一意にすることをお勧めします。

Container Sensor DaemonSet のデプロイ

cssensor-ds.yml **ファイルを作成したら** 、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します:

```
kubectl create -f cssensor-ds.yml
```

コンテナ センサー DaemonSet の削除

Qualys Container Sensor をアンインストールする必要がある場合は、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-ds.yml
```

Container Sensor DaemonSet のアップグレード

Kubernetes マスターで次の手順を実行して、Container Sensor DaemonSet を新しいバージョンに更新します。

注: Container Sensor DaemonSet が Kubernetes 環境で実行されていることを確認します。

1） Container Sensor DaemonSetkubectl の名前を取得します。
```
 get ds -n kube-system
```
2) 新しい qualys/sensor イメージを使用するように DaemonSet のイメージを更新します。
```
kubectl set image ds/<daemonset-name> -n kube-system
<containername>=<container-new-image>
```
3) ロールアウトの状態を監視する - ノードの数によっては時間がかかる場合があります。
```
kubectl rollout status daemonset <daemonset-name> -n kube-system
```
4) ロールアウト中に問題が発生した場合は、DaemonSet をロールバックして最後のイメージを
使用します。
```
kubectl rollout undo daemonset <daemonset-name> -n kube-system
```

Docker-in-Docker 環境への CI/CD センサーのインストール

このセクションでは、Docker-in-Docker 環境の CI/CD パイプラインビルドに CS センサーをインストールする方法について説明します。これにより、Docker-inDocker コンテナ内のイメージをスキャンできます。

ステップ 1: CS センサー イメージを Docker-in-Docker コンテナ内に配置する

これを行うには、1) レジストリから CS センサー イメージをプルし、コンテナがスピンアップされたときにセンサーを起動する方法と、2) CS センサー tar を含む Docker-in-Docker コンテナ イメージをベイクする方法の 2 つがあります。

レジストリから CS センサー イメージをプルし、センサーを起動します

利点：

- CS センサー イメージ/tar を使用して事前にベイクされた Docker-in-Docker コンテナ イメージを用意する必要はありません。 Docker ハブ レジストリでホストされている CS センサー イメージを簡単に使用できます

欠点：

- すべての Docker-in-Docker コンテナは、レジストリにアクセスできる必要があります。
- Docker-in-Docker コンテナがスピンアップされるたびにイメージがプルされ、オーバーヘッドになります。

CS センサー tar を含む事前にベイクされた Docker-in-Docker コンテナ イメージ

利点：

- Docker-in-Docker コンテナからレジストリにアクセスする必要はありません。
- いくつかのコマンドと installsensor.sh スクリプトを実行するだけで、CS センサーを起動できます。

欠点：

- Docker-in-Docker コンテナイメージのサイズが大きくなります。
- 新しいセンサーのリリースごとに Docker-in-Docker イメージを再ベイクする必要があります。

ステップ 2: コンテナセキュリティセンサーを起動する

これを行うには、1) Docker-in-Docker コンテナの起動時にセンサーを起動する方法と、2) ビルドジョブからセンサーを起動する方法の 2 つがあります。両方の方法について説明します。

Docker-in-Docker コンテナの起動時にセンサーを起動する

利点：

- CS センサーを起動するためにビルド パイプラインの構成を変更する必要はありません。

欠点：

- 資格情報 (アクティベーション ID/顧客 ID) は、init スクリプトに保存する必要があります。

- 事前定義された永続ストレージ・パスのみを指定できます。

Docker-in-Docker コンテナ内で init スクリプトを起動します

init スクリプトを使用してセンサーを起動します。init スクリプトには、次のコマンドがあります。

```
docker run -d --restart on-failure --cpus=0.2 -v
/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data -v
/var/run/docker.sock:/var/run/docker.sock:ro - v
/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD
URL> --net=host --name qualys-container-sensor <Qualys CS Sensor image
name from registry> --scan-thread-pool-size 4 --cicd-deployed-sensor
```

installsensor.sh スクリプトを使用する

installsensor.sh スクリプトを使用して、Docker-in-Docker の起動時にセンサーを起動します。

```
tar -xvf QualysContainerSensor.tar.xz

docker load -i qualys-sensor.tar

./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id>
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data
--cicd-deployed-sensor -s
```

ビルド ジョブからセンサーを起動する

利点：

- 資格情報 (AI/CI) とセンサー パラメーターは、ビルド ジョブ構成から渡すことができます。
- 永続ストレージは、起動時に定義できます。
- ジョブごとに一意のディレクトリを作成し (ジョブ ID を使用)、それを永続ストレージとして使用するのは簡単です。

欠点：

- CS センサーを起動するには、ビルド ジョブの構成を変更する必要があります。

docker run コマンドを使用して CS センサーを起動し、レジストリからイメージをプルします。

docker run コマンドを使用して CS センサーを起動し、レジストリから CS センサー イメージを取得します。

```
docker run -d --restart on-failure --cpus=0.2 -v
/etc/qualys:/usr/local/qualys/qpa/data/conf/agent-data -v
/var/run/docker.sock:/var/run/docker.sock:ro -v
/usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e
ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD
URL> --net=host --name qualys-container-sensor <Qualys CS Sensor image
name from registry> --scan-thread-pool-size 4 --cicd-deployed-sensor
```

事前に作成された Docker-in-Docker イメージを使用して、ビルド ジョブの一部として CS センサーを起動する。

このコマンドは、CS センサーの tar を含む事前に作成された Dockerin-Docker コンテナ イメージを使用して、ビルド ジョブの一部として CS センサーを起動します。ジョブの一部として CS センサーが起動します。

```
<path>/installsensor.sh ActivationId=<Activation id>
CustomerId=<Customer id> HostIdSearchDir=/private/etc/qualys
Storage=/tmp/qualys/sensor/data  --cicd-deployed-sensor -s
```

Docker-in-Docker ビルド コンテナで実行されている CS センサーの永続ストレージ

CS センサーの障害またはコンテナ イメージのスキャンが失敗した場合にログを取得できるように、CS センサーに適切な永続ストレージを提供してください。

Kubernetes でのセンサーのデプロイ

このセクションでは、コンテナ センサーを Kubernetes にデプロイする手順について説明します。 各セクションにジャンプします。

Kubernetes クラスタ環境でコンテナランタイムを検出する方法

センサー イメージをインストールする前に、Kubernetes クラスター環境にインストールされているコンテナ ランタイムを把握しておくことが重要です。これを知ることは、環境内でどのような種類のセンサーを実行する必要があるかを判断するのに役立ちます。

コンテナ ランタイムの詳細については、次のコマンドを実行します。

```
kubectl get nodes -o wide
```

このコマンドを使用して、名前、ステータス、ロール、経過時間、バージョン、内部および外部 IP アドレス、OS イメージ、カーネル バージョンなど、各ノードの詳細を含むクラスターに関する追加情報を取得します。以下の例を参照してください。

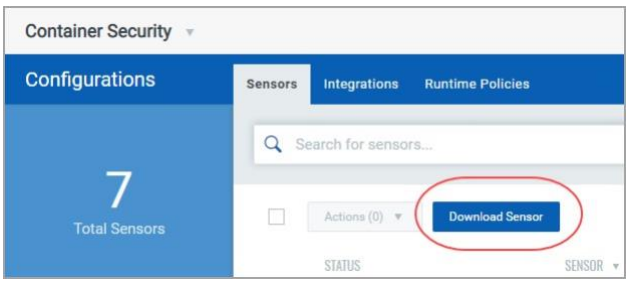
```
[root@CentOS7-Server ~]#
[root@CentOS7-Server ~]# kubectl get nodes -o wide
NAME                                    STATUS   ROLES    AGE   VERSION           INTERNAL-IP    EXTERNAL-IP    OS-IMAGE                             KERNEL-VERSION   CONTAINER-RUNTIME
gke                                     -fk8d   Ready    <none>   10d   v1.20.10-gke.1600                                Container-Optimized OS from Google   5.4.120+         containerd://1.4.4
gke                                     -h3vc   Ready    <none>   10d   v1.20.10-gke.1600                                Container-Optimized OS from Google   5.4.120+         containerd://1.4.4
gke                                     -sshl   Ready    <none>   10d   v1.20.10-gke.1600                                Container-Optimized OS from Google   5.4.120+         containerd://1.4.4
[root@CentOS7-Server ~]#
```

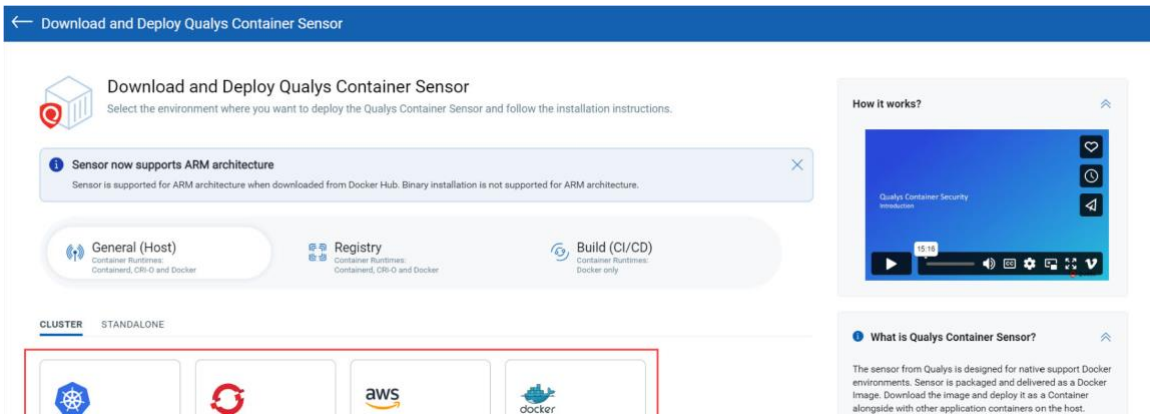
コンテナ センサー イメージの取得

Kubernetes デプロイの最初のステップは、センサー イメージを取得することです。QualysContainerSensor.tar.xz は、コンテナ セキュリティ UI からダウンロードできます。または、Docker Hub から最新の Qualys Container Sensor イメージ (qualys/qcs-sensor:latest) を使用することもできます。

UI からダウンロード

Docker がインストールされている Linux コンピューターの UI からセンサーをダウンロードします。





センサー デプロイ テンプレートは、UI からダウンロードするか、GitHub から直接 https://github.com/Qualys/cs_sensor からダウンロードできます。センサー テンプレートの使用方法の詳細については、次のセクションで説明するデプロイ手順に従います。

Orchestration Platform	Container Runtime	Sensor Template
AWS ECS	Docker	cssensor-aws-ecs.json
Kubernetes (Cloud, On-Prem)	Docker	cssensor-ds.yml

Kubernetes (Cloud, On-Prem)	Docker	cssensor-ds_pv_pvc.yml
Kubernetes (Cloud, On-Prem)	Containerd	cssensor-containerd-ds.yml
Kubernetes (Cloud, On-Prem)	CRI-O	cssensor-crio-ds.yml
Docker Swarm	Docker	cssensor-swarm-ds.yml
Red Hat OpenShift	Docker	cssensor-openshift-ds.yml
Red Hat OpenShift	CRI-O	cssensor-openshift-crio-ds.yml

注: CRI-O ランタイムは、一般 (ホスト) センサーとレジストリ センサーでサポートされています。ビルド CI/CD センサーではサポートされていません。

ファイルをダウンロードしたら、次のコマンドを使用してセンサー パッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Docker ランタイム環境でイメージをロードするには、次のようにします。

次のコマンドを使用して、Qualys センサー イメージを Kubernetes クラスタ内のすべてのノードに共通のリポジトリにプッシュします。

```
sudo docker load -i qualys-sensor.tar
sudo docker tag <IMAGE NAME/ID> <URL to push image to repository>
sudo docker push <URL to push image to the repository>
```

例えば：

```
sudo docker load -i qualys-sensor.tar sudo docker tag
c3fa63a818df mycloudregistry.com/container-
sensor:qualys-sensor-xxx
sudo docker push mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

注: これらの例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

Containerd ランタイム環境にイメージをロードするには、次のようにします。

次のコマンドを使用して、Qualys センサー イメージを Kubernetes クラスタ内のすべてのノードに共通のリポジトリにプッシュします。

```
ctr -n=k8s.io images import qualys-sensor.tar ctr images tag <IMAGE
NAME/ID> <URL to push image to the repository> ctr images push <URL
to push image to the repository>
```

例えば：

```
ctr -n=k8s.io images import qualys-sensor.tar ctr images tag
c3fa63a818df mycloudregistry.com/container-sensor:qualyssensor-xxx
ctr images push mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

注:これらの例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

CRI-O ランタイム環境でイメージをロードするには、次のようにします。

次のコマンドを使用して、Qualys センサー イメージを Kubernetes クラスタ内のすべてのノードに共通のリポジトリにプッシュします。

```
podman load -i qualys-sensor.tar podman tag <IMAGE NAME/ID> <URL
to push image to the repository> podman push <URL to push image
to the repository>
```

例えば：

```
podman load -i qualys-sensor.tar podman tag c3fa63a818df
mycloudregistry.com/container-sensor:qualyssensor-xxx
podman push mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

注: これらの例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

Docker Hub からイメージを取得する

Docker Hub の最新の Qualys Container Sensor イメージ qualys/qcs-sensor:latest を使用します。Docker Hub の Container Security Sensor は、次のように使用できます。

Qualys/QCS センサー:<Day>

Qualys/QCS センサー:Latest

Docker Hub で最新のタグを検索します。Docker Hub Qualys Container Sensor イメージは、プライベート レジストリにプッシュすることも、直接使用することもできます。すべての Kubernetes ノードから、Docker Hub/プライベート レジストリ (CS センサー イメージが公開されている場所) にアクセスできることを確認します。

Azure Kubernetes Service (AKS) にデプロイする

Azure Kubernetes Service (AKS) クラスターにセンサーをデプロイする手順は、Kubernetes のバージョンとコンテナ ランタイムによって異なります。

Kubernetes バージョン 1.18 以前を使用して AKS クラスターにセンサーをデプロイし、コンテナ ランタイムが Docker ランタイムの場合は、「Kubernetes でのデプロイ - Docker ランタイム」を参照してください。

Kubernetes バージョン 1.19 を使用して AKS クラスターにセンサーをデプロイし、コンテナ ランタイムが Containerd ランタイムの場合は、「Kubernetes でのデプロイ - Containerd ランタイム」を参照してください。

Kubernetes へのデプロイ - Docker ランタイム

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

他のアプリケーション コンテナと同様に Container Sensor を DaemonSet に統合し、レプリケーション係数を 1 に設定して、Docker ホストに常にセンサーがデプロイされるようにします。この情報は、Amazon Elastic Container Service for Kubernetes (Amazon EKS)、Google Kubernetes Engine (GKE)、および Azure Kubernetes Service (AKS) に適用されます。

Kubernetes にデプロイする Qualys センサーの DaemonSet を作成するには、次の手順を実行します。

注: Container Sensor に、永続ストレージと Docker デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

cssensor-ds.yml ファイルを変更する

以下は、 Qualys コンテナ センサー。お客様の便宜のために、このテンプレートは QualysContainerSensor.tar.xz で cssensor-ds.yml ファイルとして入手できます。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

重要: yaml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

```
kind: List
apiVersion: v1

items:
- kind: Namespace
apiVersion: v1
metadata:
name: qualys
# Service Account
- kind: ServiceAccount
apiVersion: v1
metadata:
name: qualys-service-account
namespace: qualys
# Role for read/write/delete permission to qualys namespace
- kind: Role
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

```
name: qualys-reader-role
namespace: qualys
rules:
- apiGroups: ["","batch"]
resources: ["pods","jobs"]
verbs: ["get", "list", "watch","create", "delete",
"deletecollection"]
- apiGroups: [""]
resources: ["pods/status"]
verbs: ["get"]
- apiGroups: [""]
resources: ["pods/attach", "pods/exec"]
verbs: ["create"]
- kind: ClusterRole
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-cluster-reader-role
rules:
- apiGroups: [""]
resources: ["nodes", "pods/status",
"replicationcontrollers/status", "nodes/status"]
verbs: ["get"]
- apiGroups: ["apps"]
resources: ["replicasets/status", "daemonsets/status",
"deployments/status", "statefulsets/status"]
verbs: ["get"]
- apiGroups: ["batch"]
resources: ["jobs/status", "cronjobs/status"]
verbs: ["get"]
# RoleBinding to assign permissions in qualys-reader-role to qualys-
service-account
- kind: RoleBinding
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-reader-role-rb
namespace: qualys
subjects:
- kind: ServiceAccount
name: qualys-service-account
namespace: qualys
roleRef:
kind: Role
name: qualys-reader-role
apiGroup: rbac.authorization.k8s.io
- kind: ClusterRoleBinding
# if k8s version is 1.17 and earlier then change apiVersion to
"rbac.authorization.k8s.io/v1beta1"
```

```
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qualys-cluster-reader-rb
subjects:
- kind: ServiceAccount
name: qualys-service-account
namespace: qualys
roleRef:
kind: ClusterRole
name: qualys-cluster-reader-role
apiGroup: rbac.authorization.k8s.io
# Qualys Container Sensor pod with
- apiVersion: apps/v1
kind: DaemonSet
metadata:
name: qualys-container-sensor
namespace: qualys
labels:
k8s-app: qualys-cs-sensor
spec:
selector:
matchLabels:
name: qualys-container-sensor
updateStrategy:

type: RollingUpdate
template:
metadata:
labels:
name: qualys-container-sensor
spec:
#tolerations:
# this toleration is to have the daemonset runnable on master
nodes
# remove it if want your masters to run sensor pod
#- key: node-role.kubernetes.io/master
# effect: NoSchedule
serviceAccountName: qualys-service-account
containers:
- name: qualys-container-sensor
image: qualys/qcs-sensor:latest
imagePullPolicy : IfNotPresent
resources:
limits:
cpu: "0.2" # Default CPU usage limit on each node for
sensor.
args: ["--k8s-mode"]
env:
- name: CUSTOMERID
value: __customerId
- name: ACTIVATIONID
value: __activationId
```

```
- name: POD_URL
value:
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
value: "10"
# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- name: DOCKER_TLS_VERIFY
# value: "1"
# uncomment(and indent properly) below section if proxy is required to
connect Qualys Cloud
#- name: qualys_https_proxy
# value: <proxy FQDN or Ip address>:<port#>
- name: QUALYS_POD_NAME
valueFrom:
fieldRef:
fieldPath: metadata.name
- name: QUALYS_POD_NAMESPACE
valueFrom:
fieldRef:

fieldPath: metadata.namespace
volumeMounts:
- mountPath: /var/run/docker.sock
name: socket-volume
readOnly: true
- mountPath: /usr/local/qualys/qpa/data
name: persistent-volume
- mountPath: /usr/local/qualys/qpa/data/conf/agent-data
name: agent-volume
# uncomment(and indent properly) below section if proxy(with CA cert)
required to connect Qualys Cloud
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
# name: proxy-cert-path
# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- mountPath: /root/.docker
# name: tls-cert-path
securityContext:
allowPrivilegeEscalation: false
volumes:
- name: socket-volume
hostPath:
path: /var/run/docker.sock
type: Socket
- name: persistent-volume
hostPath:
path: /usr/local/qualys/sensor/data
type: DirectoryOrCreate
- name: agent-volume
hostPath:
path: /etc/qualys
type: DirectoryOrCreate
```

```
# uncomment(and indent properly) below section if proxy(with CA cert)
required to connect Qualys Cloud
#- name: proxy-cert-path
# hostPath:
# path: <proxy certificate path>
# type: File
# uncomment(and indent properly) below section if using Docker HTTP
socket with TLS
#- name: tls-cert-path
# hostPath:
# path: <Path of directory of client certificates>
# type: Directory
hostNetwork: true
```

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切な権限を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、まず他の名前空間から Qualys Container Sensor をアンインストールしてから、「qualys」名前空間に新たにデプロイする必要があります。

get、list、watch - のリソースを監視するには次のアクセス許可が必要です: 'qualys' 名前空間

create、delete、deleteCollection - イメージの脆弱性評価用のコンテナーを 'Qualys' 名前空間に生成し、その後にクリーンアップします。

yaml ファイルのパラメーターを変更する

cssensor-ds.yml ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

spec の **tolerations** セクション**のコメントを解除し** て、Sensor daemonset をマスターノードにデプロイします。

```
spec:
      #tolerations:
      # this toleration is to have the daemonset runnable on master
      nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule

containers:
      - name: qualys-container-sensor   image: <CS Sensor image name in
      the private/docker hub registry>   args: ["--k8s-mode"]
```

注: センサー ポッドを実行するすべてのノードが、センサー イメージが格納されているプライベート レジストリまたは Docker ハブ レジストリにアクセスできることを確認してください。

CI/CD 環境用にセンサーをデプロイする場合は、args 値を次のように指定します。

```
args: ["--k8s-mode","--cicd-deployed-sensor"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
args: ["--k8s-mode","--registry-sensor"]
```

コンソールでログを出力する場合は、args の追加値として "**--enable-console-logs**" を指定します。

ログレベルを変更する場合は、args の追加値として「**--log-level**」、「<0〜5>の数値」を指定します。

```
args: ["--k8s-mode", "--log-level", "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "**--scan-thread-pool-size**"、"<number of threads>" を指定します。

```
args: ["--k8s-mode", "--scan-thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数を定義する場合は、args の追加値として "--log-filepurgecount", "<digit>" を指定します。デフォルトの "**--logfilepurgecount**", "5" は config で適用されます。log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "--log-filesize"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "--log-filepurgecount"、"<digit>" を指定します。既定値は "**--log-filesize**": "10M" と "--log-filepurgecount": "5" で、config を介して適用されます。

```
args: ["--k8s-mode", "--log-filesize", "5M", "--
logfilepurgecount", "4"]
```

kubernetes ネイティブの 'kubectl run' コマンドを使用してイメージ スキャン ポッドをインスタンス化する場合は、args の追加値として "--use-kubectl" を指定します。この場合、センサーはネイティブの kubernetes 機能を使用してイメージ スキャンを開始します。この引数を省略すると、イメージ コンテナは 'docker run' を使用して起動されます。

```
args: ["--k8s-mode", "--use-kubectl"]
```

TLS 認証が有効になっている場合は、args に Docker クライアント証明書、クライアント秘密鍵、および CA 証明書の名前を指定します

```
args: ["--k8s-mode", "--tls-cacert","<file name of the CA
certificate that was used to sign docker server certificate>", "-
tls-cert", "<docker client certificate file name>", "--tls-key",
"<docker client private key file name>"]
```

注: 3 つのファイルのいずれかにそれぞれ ca.pem、cert.pem、key.pem などのデフォルト名がある場合は、対応する引数を省略できます。

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「**--mask-env-variable**」パラメータを追加します。

```
args: ["--k8s-mode", "--mask-env-variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「**--disableImageScan**」パラメーターを追加します。

```
args: ["--k8s-mode", "--disableImageScan"]
```

スキャンポリシーを指定する場合は、args に "--scanning-policy" パラメーターを追加します。スキャン ポリシーで使用できる値は、"DynamicScanningOnly"、"StaticScanningOnly"、および "DynamicWithStaticScanningAsFallback" です。

```
args: ["--k8s-mode", "--scanning-policy", "StaticScanningOnly"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「**--disable-log4j-scanning**」パラメータを追加します。

```
args: ["--k8s-mode", "--disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "**--disable-log4j-static-detection**" パラメーターを追加します。

```
args: ["--k8s-mode", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「**--optimize-image-scans**」パラメーターを追加します。

```
args: ["--k8s-mode", "--optimize-image-scans"]
```

SCA スキャン機能がある場合は、args に "**--perform-sca-scan**" パラメーターを追加することで、コンテナイメージの SCA スキャンを有効にできます。

```
args: ["--k8s-mode", "--perform-sca-scan"]
```

デフォルトでは、SCA スキャンはオンライン・モードで実行されます。SCA スキャンのインターネット・アクセスを使用不可にして、スキャンをオフライン・モードで実行することもできます。注 - SCA スキャンはオンライン・モードで実行することをお勧めします。Java のソフトウェア・パッケージ列挙の品質は、SCA スキャンがオフライン・モードで実行されると大幅に低下します。正確なパッケージ検出のために、リモートの Maven リポジトリを参照する必要がある場合があります。これは、イメージの脆弱性ポスチャの精度に影響を与える可能性があります。

```
args: ["--k8s-mode", "--perform-sca-scan" "--disallow-
internetaccess-for-sca"]
```

デフォルトの SCA スキャン・コマンド・タイムアウトは 5 分 (300 秒) です。デフォルトのタイムアウトは、秒単位で指定した新しい値で上書きできます。例えば、大きなコンテナ・イメージをスキャンするときには、SCA スキャンが完了する時間を確保するために、SCA スキャンのタイムアウトを増やす必要がある場合があります。

注: **--sca-scan-timeout-in-seconds=600** パラメータは、シークレットとマルウェアの検出にも適用されます。

```
args: ["--k8s-mode", "--perform-sca-scan", "--sca-scan-timeout-
inseconds=600"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-
malwaredetection"]
```

[リソース] で 、次の項目を指定します。

```
resources: limits: cpu: "0.2" # Default CPU usage limit on
each node for sensor.
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
  limits:
    cpu: "0.2" # Default CPU usage limit on each node for sensor
     memory: "500Mi"
     requests:
     memory: "300Mi"
```

Container Sensor にメモリリソースの値(制限またはリクエスト)のいずれかが指定され、args に「--use-kubectl」が指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required) -
      name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required) -
      name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT  value:
  "10"
```

TLS 認証を有効にするには、次の 2 行のコメントを解除します。

```
- name: DOCKER_TLS_VERIFY      value: "1"
```

注:TLS を無効にするには、DOCKER_TLS_VERIFY=""(空の文字列)を使用するか、削除するか、yml ファイルにコメントを付けたままにします。

注: TLS 経由で Docker デーモンとのセンサー通信を有効にすることで、お客様は docker 認証プラグインを使用して docker ソケットへのセンサーのアクセスを制限できます。

注:TLS 認証が有効で、DOCKER_HOST が指定されていない場合、センサーは実行されているワーカー・ノードの FQDN を自動的に検出し DOCKER_HOST センサー内の環境変数を<ワーカー・ノードの FQDN>:<2376>に設定します(2376 は Docker デーモンのデフォルトの TLS TCP ポートです)

注: 以下を追加することで、DOCKER_HOST を 127.0.0.1:<port#> または localhost:<port#> に設定できます。

```
- name: DOCKER_HOST
```

```
value: "<loopback IPv4 address or hostname>:<port#>"
```

注: DOCKER_HOST に設定されている FQDN、ホスト名、または IPv4 アドレスが、各ワーカー ノードの Docker サーバー証明書の CN またはサブジェクトの別名と一致していることを確認してください

TLS 認証を有効にする必要がある場合は、ボリュームの下の tls-cert-path のコメントを解除し、クライアント証明書のディレクトリパスを指定するか、不要な場合はそのままにしておきます。

```
#- name: tls-cert-path
#   hostPath:
#      path: <Path of directory of client certificates>
#     type: Directory
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

- `name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL  value: "30"`

TLS を使用している場合は、volumeMounts の下の部分のコメントも解除します。それ以外の場合は、コメントアウトしたままにします。

```
#- mountPath: /root/.docker
#name: tls-cert-path
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。
センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の proxy-cert-path のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
#   hostPath:
#      path: /root/cert/proxy-certificate.crt
```

```
#      type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
#  name: proxy-cert-path
```

cssensor-ds.yml ファイルを変更したら 、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します。

```
kubectl create -f cssensor-ds.yml
```

Qualys Container Sensor をアンインストールする必要がある場合は、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

永続ボリューム要求の使用

PersistentVolumeClaim(PVC)を使用して、指定した総永続ボリュームから特定のサイズのストレージを要求できます。

cssensor-ds_pv_pvc.yml ファイルを変更する

以下は cssensor-ds_pv_pvc.yml です。お客様の便宜のために、cssensords_pv_pvc.yml ファイルを QualysContainerSensor.tar.xz から抽出できます。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください。

重要: yaml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

yaml ファイルのパラメーターを変更する

cssensor-ds_pv_pvc.yml ファイルを変更して、次のパラメーターの値を指定します。yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてく

ださい。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

```
- kind: PersistentVolume
  apiVersion: v1
  metadata:
    name: qualys-sensor-pv-volume
   labels:
         type: local
  spec:
    storageClassName: manual
    capacity:
      storage: 5Gi
  accessModes:
     - ReadWriteOnce
  hostPath:
       path: "/mnt/data/"
- kind:PersistentVolumeClaim
  apiVersion: v1
  metadata:
    name: qualys-sensor-pv-claim
    namespace: qualys
  spec:
   storageClassName: manual
   accessModes:          -
   ReadWriteOnce
   resources:
    requests:
     storage: 1Gi
```

ここでは、1Gi の PVC が 5Gi の永続ボリュームに作成されます。 サポートされている永続ボリュームの種類の一覧については、 ここをクリックしてください。

ボリュームの下に PVC の名前を追加します。

```
- name: persistent-volume
  persistentVolumeClaim:
  claimName: qualys-sensor-pv-claim
```

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、cssensor-ds.yml ファイルを変更し、args の追加値として "**--sensor-without-persistent-storage**" を指定します。

```
args: ["--k8s-mode","--sensor-without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-withoutpersistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath: /usr/local/qualys/qpa/data
   name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
 - name: persistent-volume  hostPath:
    path: /usr/local/qualys/sensor/data
     type: DirectoryOrCreate
```

Kubernetes へのデプロイ - Containerd ランタイム

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

cssensor-containerd-ds.yml ファイルを変更します

Qualys コンテナ センサーのテンプレートは QualysContainerSensor.tar.xz で入手できます。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできます。

重要: yml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切なアクセス許可を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、他の名前空間から Qualys Container Sensor をアンインストールし、「qualys」名前空間に新たにデプロイする必要があります。

必要な権限：

get、list、watch - クラスター全体の脆弱性をスキャンするリソースを監視します。

作成、削除、削除コレクション - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成するには、クラスター全体でポッドをスキャンし、その後クリーンアップします。

注: コンテナ・センサーに、永続ストレージと containerd デーモン・ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

yaml ファイルのパラメーターを変更する

cssensor-containerd-ds.yml ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

ここでは、一般的なセンサーを想定しています。spec の tolerations セクションのコメントを解除して、Sensor daemonset をマスターノードにデプロイします。

```
spec:
      #tolerations:
      # this toleration is to have the daemonset runnable on master
      nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule

containers:
```

```
      - name: qualys-container-sensor
        image: <CS Sensor image name in the private/docker hub registry>
args: ["--k8s-mode", "--container-runtime", "containerd"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
args: ["--k8s-mode","--container-runtime", "containerd", "-
registry-sensor"]
```

CICD センサーをデプロイする場合は、args 値を次のように指定します。

```
args: ["--k8s-mode","--container-runtime", "containerd", "--
cicddeployed-sensor"]
```

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
loglevel",  "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
scanthread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "**--log-filesize**"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "**--log-filepurgecount**"、"<digit>" を指定します。既定値は "--log-filesize": "10M" と "**--log-filepurgecount**": "5" で、config を介して適用されます。

"**--log-filesize**": ログファイルあたりの最大サイズを定義するために使用できます。たとえば、"10K" (キロバイト)、"10M" (メガバイト)、"10" (バイト) などです。

"**--log-filepurgecount**":生成されるアーカイブログファイルの数を定義するために使用できますが、log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
logfilesize", "5M", "--log-filepurgecount", "4"]
```

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「**--mask-env-variable**」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
maskenv-variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「**--disableImageScan**」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
disableImageScan"]
```

スキャンポリシーを指定する場合は、args に "**--scanning-policy**" パラメーターを追加します。スキャン ポリシーで使用できる値は、"DynamicScanningOnly"、"StaticScanningOnly"、および "DynamicWithStaticScanningAsFallback" です。

```
args: ["--k8s-mode", "--scanning-policy", "StaticScanningOnly"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「**--disable-log4j-scanning**」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "**--disable-log4j-static-detection**" パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「**--optimize-image-scans**」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
optimize-image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "**--perform-secretdetection**" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "**--performmalware-detection**" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-
malwaredetection"]
```

[リソース] で 、次の項目を指定します。

```
resources: limits: cpu: 0.2 # Default CPU usage limit (20% of
one core on the host).
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
 limits:
  cpu: "0.2" # Default CPU usage limit on each node for sensor
   memory: "500Mi"      requests:        memory: "300Mi"
```

Container Sensor にメモリリソース値(制限または要求)のいずれかが指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required)
      name: ACTIVATIONID
      value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required) -
      name: CUSTOMERID
      value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後

にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL  value: "30"
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の proxy-cert-path のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
#  hostPath:
#      path: /root/cert/proxy-certificate.crt
#      type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
#  name: proxy-cert-path
```

Container Sensor DaemonSet をデプロイする方法

cssensor-containerd-ds.yml ファイルを変更したら 、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します。

```
kubectl create -f cssensor-containerd-ds.yml
```

Container Sensor DaemonSet を削除する方法

Qualys Container Sensor をアンインストールするには、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-containerd-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、cssensor-containerd-ds.yml ファイルを変更し、args の追加値として "**--sensor-without-persistent-storage**" を指定します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-
sensor-without-persistent-storage"]
```

ログを保持するには、"**--enable-console-logs**" オプションと "**--sensor-withoutpersistent-storage**" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath:
/usr/local/qualys/qpa/data  name:
persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume  hostPath:
    path: /usr/local/qualys/sensor/data
type: DirectoryOrCreate
```

How to Tag Target Images for the CICD Sensor

CICD イメージに qualys_scan_target:<tag> のタグを付けてスキャン対象としてマークするには、**nerdctl ツール**を使用します。画像に qualys_scan_target:<anytag> でタグを付けるには、<image-repo:tag> を使用します。

```
nerdctl -n k8s.io tag <image-repo:tag>
qualys_scan_target:<anytag>
```

例えば

```
./nerdctl -n k8s.io tag docker.io/library/known:latest
qualys_scan_target:known
```

メモ:

- nerdctl バイナリは、ホスト上で使用可能である必要があります。
- ターゲットイメージは、k8s.io 名前空間にのみ作成する必要があります。

Kubernetes へのデプロイ - CRI-O ランタイム

注: OpenShift に CRI-O ランタイムがある場合は、「Kubernetes へのデプロイ - CRI-O ランタイムを使用した OpenShift4.4+」の手順に従ってください。

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

注: CRI-O ランタイムは、一般 (ホスト) センサーとレジストリー・センサーでサポートされています。ビルド CI/CD センサーではサポートされていません。

cssensor-crio-ds.yml ファイルを変更する

Qualys コンテナセンサーのテンプレートは https://github.com/Qualys/cs_sensor から直接 QualysContainerSensor.tar.xz を入手下さい。

重要: yml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切なアクセス許可を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、他の名前空間から Qualys Container Sensor をアンインストールし、「qualys」名前空間に新たにデプロイする必要があります。

次の権限が必要です。

get、list、watch - クラスター全体の脆弱性をスキャンするリソースを監視します。

作成、削除、削除コレクション - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成するには、クラスター全体でポッドをスキャンし、その後クリーンアップします

注: コンテナ センサーに、永続ストレージと cri-o デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

yaml ファイルのパラメーターを変更する

cssensor-crio-ds.ym ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

ここでは、一般的なセンサーを想定しています。前述したように、CRI-O は一般 (ホスト) センサーとレジストリ センサーでサポートされています。ビルド CI/CD センサーではサポートされていません。

spec の tolerations セクションのコメントを解除して、Sensor daemonset をマスターノードにデプロイします。

```
spec:
```

```
      #tolerations:
      # this toleration is to have the daemonset runnable on master
      nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule
containers:
      - name: qualys-container-sensor
      image: qualys/qcs-sensor:latest
      imagePullPolicy: IfNotPresent
        args: ["--k8s-mode", "--container-runtime", "cri-o"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--container-runtime", "cri-o", "—-registry-
      sensor"]
```

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "—-log-
      level",  "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "—scan-
      thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "--log-filesize"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "--log-filepurgecount"、"<digit>" を指定します。既定値は "--log-filesize": "10M" と "--log-filepurgecount": "5" で、config を介して適用されます。

"**--log-filesize**": ログファイルあたりの最大サイズを定義するために使用できます。たとえば、"10K" (キロバイト)、"10M" (メガバイト)、"10" (バイト) などです。

"**--log-filepurgecount**":生成されるアーカイブログファイルの数を定義するために使用できますが、log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "—log-
      filesize", "5M", "--log-filepurgecount", "4"]
```

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「**--mask-env-variable**」パラメータを追加します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "--mask-env-
      variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「--disableImageScan」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "--
disableImageScan"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「--disable-log4j-scanning」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "--
disablelog4j-scanning"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「--optimize-image-scans」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "—optimize-
image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-malware-
detection"]
```

[リソース] で 、次の項目を指定します。

```
resources:   limits:     cpu: "0.2" # Default CPU usage limit on
each node for sensor.
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
   resources:
     limits:
     cpu: "0.2" # Default CPU usage limit on each node for sensor
      memory: "500Mi"
    requests:
     memory: "300Mi"
```

Container Sensor にメモリリソース値(制限または要求)のいずれかが指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required) -
      - name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required) -
      - name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- Name: POD_URL  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
-name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
 value: "30"
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の **proxy-cert-path** のコメントを解除するか、不要な場合はそのままにします。

```
# uncomment (and indent properly) below section if proxy (with CA
cert) is required to connect to the Qualys Cloud Platform
      #- name: proxy-cert-path
      #  hostPath:
      #      path: <proxy certificate path>
      #      type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
      #- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
      #   name: proxy-cert-path
```

Container Sensor DaemonSet をデプロイする方法

cssensor-crio-ds.yml ファイルを変更したら 、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します。

```
kubectl create -f cssensor-crio-ds.yml
```

Container Sensor DaemonSet を削除する方法

Qualys Container Sensor をアンインストールするには、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-crio-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。永続ストレージなしでセンサーを起動するには、cssensor-crio-ds.yml ファイルを変更し、args の追加値として "--sensor-without-persistent-storage" を指定します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "—sensor-
without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-withoutpersistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath:/usr/local/qualys/qpa/data
name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume
   hostPath:
    path: /usr/local/qualys/sensor/data
     type: DirectoryOrCreate
```

Kubernetes へのデプロイ - OpenShift

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

コンテナセンサーを他のアプリケーションコンテナと同様に DaemonSet に統合して、Docker ホストに常にセンサーがデプロイされるようにします。OpenShift にデプロイする Qualys センサーの DaemonSet を作成するには、以下のステップを実行します。

注: Container Sensor に、永続ストレージと Docker デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

OpenShift マスターの Qualys クラウドポータルから **QualysContainerSensor.tar.xz** ファイルをダウンロードします。

センサーパッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

以下のコマンドを使用して、OpenShift クラスター内のすべてのノードに共通するリポジトリーに Qualys センサーイメージをプッシュします。

```
sudo docker load -i qualys-sensor.tar
sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
sudo docker push <URL to push image to the repository>
```

例えば：

```
sudo docker load -i qualys-sensor.tar
sudo docker tag c3fa63a818df
mycloudregistry.com/containersensor:qualys-sensor-xxx
sudo docker push mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

注: 例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

cssensor-openshift-ds.yml ファイルを変更します

cssensor-openshift-ds.yml ファイル (QualysContainerSensor.tar.xz から抽出) を変更して、以下のパラメーターの値を指定します。yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

```
    serviceAccountName:
          qualysuser
```

serviceAccountName がポッド宣言で指定されていることを確認します。

```
  containers:
      - name: qualys-container-sensor
      image: <CS Sensor image name in the docker hub/private registry>
      securityContext:
      privileged: true
      args: ["--k8s-mode"]
```

CI/CD 環境用にセンサーをデプロイする場合は、**args** 値を次のように指定します。

```
    args: ["--k8s-mode","--cicd-deployed-sensor","--log-level","5","-
    log-filesize","5M","--log-filepurgecount","4"]
```

レジストリー・センサーをデプロイする場合は、**args** 値を次のように指定します。

```
    args: ["--k8s-mode","--registry-sensor","--log-level","5","--
    logfilesize","5M","--log-filepurgecount","4"]
```

注: 上記の引数の "**--log-level**"、"**--log-filesize**"、および "**--log-filepurgecount**" の値は サンプルにすぎません。必要に応じて適切な値を指定します。

コンソールでログを出力する場合は、args の追加値として "--enable-console-logs" を指定します。

CPU 使用率を特定の値に制限するには、次のように変更します。 (オプション) 「リソース」で、次の項目を指定します。

```
resources:
     limits:
      cpu: "0.2" # Default CPU usage limit(20% of one core on the host).
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。

たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

env で 、次の項目を指定します。

アクティベーション ID (必須)

```
- name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

お客様 ID (必須)

```
- name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

プロキシ情報を指定するか、不要な場合は削除します。

```
- name: qualys_https_proxy
  value: proxy.localnet.com:3128
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

  ```
  #- name: qualys_https_proxy
  #   value: <proxy FQDN or Ip address>:<port#>
  ```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

  ```
  #- name: https_proxy
  #  value: <proxy FQDN or Ip address>:<port#>
  ```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
  value: "30"
```

ボリュームでプロキシ証明書のパスを指定するか、不要な場合は削除します。

```
- name: proxy-cert-path
  hostPath:
       path: /root/cert/proxy-certificate.crt
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。

プロキシを使用している場合は、センサーが Container Management Server と通信するための有効な証明書ファイルがすべての OpenShift ノードにあることを確認します。

プロキシを使用しておらず、上記の部分を削除した場合は、volumeMounts から次の部分も削除できます 。

```
- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
  name: proxy-cert-path
```

cssensor-openshift-ds.yml ファイルを変更したら 、OpenShift マスターで以下のコマンドを実行して DaemonSet を作成します。

```
oc create -f cssensor-openshift-ds.yml
```

Qualys Container Sensor をアンインストールする必要がある場合は、OpenShift マスターで以下のコマンドを実行します。

```
oc delete ds qualys-container-sensor -n kube-system
```

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、cssensor-openshift-ds.yml ファイルを変更し、args の追加値として「**--sensor-without-persistent-storage**」を指定します。

```
args: ["--k8s-mode","--sensor-without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
 - mountPath: /usr/local/qualys/qpa/data
   name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume  hostPath:
    path: /usr/local/qualys/sensor/data
```

Kubernetes へのデプロイ - CRI-O ランタイムを使用した OpenShift4.4+

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

注: CRI-O ランタイムは、一般 (ホスト) センサーとレジストリー・センサーでサポートされています。ビルド CI/CD センサーではサポートされていません。

cssensor-openshift-crio-ds.yml ファイルの変更

Qualys コンテナセンサーのテンプレートは https://github.com/Qualys/cs_sensor から直接 QualysContainerSensor.tar.xz を入手下さい。

重要: yml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切な権限を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBindings は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。

次の権限が必要です。

get、list、watch - クラスター全体の脆弱性をスキャンするリソースを監視します。

作成、削除、削除コレクション - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成するには、クラスター全体でポッドをスキャンし、その後クリーンアップします

注: コンテナ センサーに、永続ストレージと cri-o デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

yaml ファイルのパラメーターを変更する

cssensor-openshift-crio-ds.yml ファイルを Openshift4.4 クラスターのマスターノードにコピーし、以下のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

ここでは、一般的なセンサーを想定しています。前述したように、CRI-O は一般 (ホスト) センサーとレジストリ センサーでサポートされています。ビルド CI/CD センサーではサポートされていません。

spec の tolerations セクションのコメントを解除して、Sensor daemonset をマスターノードにデプロイします

```
spec:
      #tolerations:
      # this toleration is to have the daemonset runnable on master nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule
```

```
containers:
      - name: qualys-container-sensor
      image: <CS Sensor image name in the private/docker hub registry>
      args: ["--k8s-mode", "--container-runtime", "cri-o"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--container-runtime", "cri-o", "—-registry-
      sensor"]
```

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "—-log-
      level", "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "—scan-
      thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "--log-filesize"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "--log-filepurgecount"、"<digit>" を指定します。既定値は "--log-filesize": "10M" と "--log-filepurgecount": "5" で、config を介して適用されます。

"--log-filesize": ログファイルあたりの最大サイズを定義するために使用できます。たとえば、"10K" (キロバイト)、"10M" (メガバイト)、"10" (バイト) などです。

"--log-filepurgecount":生成されるアーカイブログファイルの数を定義するために使用できますが、log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。 `args: ["--k8s-mode", "--container-runtime", "cri-o", "--logfilesize", "5M", "--log-filepurgecount", "4"]`

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「--mask-env-variable」パラメータを追加します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "--mask-env-
      variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「--disableImageScan」パラメーターを追加します。

```
      args: ["--k8s-mode", "--container-runtime", "cri-o", "-
      disableImageScan"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「--disable-log4j-scanning」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "—disable-log4j-scanning"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「--optimize-image-scans」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "—optimize-image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "-perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-malware-detection"]
```

[リソース] で 、次の項目を指定します。

```
resources:
limits:
cpu: "0.2" # Default CPU usage limit on each node for sensor.
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
 limits:
```

```
      cpu: "0.2" # Default CPU usage limit on each node for sensor
      memory: "500Mi"
    requests:
      memory: "300Mi"
```

Container Sensor にメモリリソース値(制限または要求)のいずれかが指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required)
- name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required)
- name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL
  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
  value: "30"
```

プロキシ情報を指定するか、不要な場合は削除します。

```
- name: qualys_https_proxy
  value: <PROXY_URL>
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の proxy-cert-path のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
#   hostPath:
#       path: /root/cert/proxy-certificate.crt
#       type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
#   name: proxy-cert-path
```

Container Sensor DaemonSet をデプロイする方法

cssensor-openshift-crio-ds.yml ファイルを変更したら 、OpenShift マスターで以下のコマンドを実行して DaemonSet を作成します。

```
oc create -f cssensor-openshift-crio-ds.yml
```

Container Sensor DaemonSet を削除する方法

Qualys Container Sensor をアンインストールするには、Openshift マスターで次のコマンドを実行します。

```
oc delete -f cssensor-openshift-crio-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、cssensor-openshift-crio-ds.yml ファイルを変更し、args の追加値として「--sensor-without-persistent-storage」を指定します。

```
args: ["--k8s-mode", "--container-runtime", "cri-o", "—sensor-
without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath: /usr/local/qualys/qpa/data  name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume  hostPath:
   path: /usr/local/qualys/sensor/data
    type: DirectoryOrCreate
```

TKGI を使用した Kubernetes へのデプロイ - Docker ランタイム

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

Kubernetes にデプロイする Qualys センサーの DaemonSet を作成するには、次の手順を実行します。

注: Container Sensor に、永続ストレージと Docker デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

cssensor-ds.yml ファイルを変更する

以下は、 Qualys コンテナ センサー。お客様の便宜のために、このテンプレートは QualysContainerSensor.tar.xz で cssensor-ds.yml ファイルとして入手できます。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください。

重要: yaml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず尊重してください。

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切な権限を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、まず他の名前空間から Qualys Container Sensor をアンインストールしてから、「qualys」名前空間に新たにデプロイする必要があります。

次の権限が必要です: get、list、watch - 'qualys' 名前空間のリソースを監視するため

create、delete、deleteCollection - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成し、その後にクリーンアップします。

yaml ファイルのパラメーターを変更する

cssensor-ds.yml ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yaml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

spec の **tolerations** セクション**のコメントを解除し**て、Sensor daemonset をマスターノードにデプロイします。

```
spec:
      #tolerations:
      # this toleration is to have the daemonset runnable on master nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule

containers:
      - name: qualys-container-sensor
        image: <CS Sensor image name in the private/docker hub registry>
        args: ["--k8s-mode"]
```

注: センサー ポッドを実行するすべてのノードが、センサー イメージが格納されているプライベート レジストリまたは Docker ハブ レジストリにアクセスできることを確認してください。

CI/CD 環境用にセンサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--cicd-deployed-sensor"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--registry-sensor"]
```

コンソールでログを出力する場合は、args の追加値として "--enable-console-logs" を指定します。

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
args: ["--k8s-mode", "--log-level", "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
args: ["--k8s-mode", "--scan-thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数を定義する場合は、args の追加値として "--log-filepurgecount", "<digit>" を指定します。デフォルトの "--logfilepurgecount", "5" は config で適用されます。log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "--log-filesize"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "--log-filepurgecount"、"<digit>" を指定します。既定値は "--log-filesize": "10M" と "--log-filepurgecount": "5" で、config を介して適用されます。

```
args: ["--k8s-mode", "--log-filesize", "5M", "—log-
filepurgecount", "4"]
```

kubernetes ネイティブの 'kubectl run' コマンドを使用してイメージ スキャン ポッドをインスタンス化する場合は、args の追加値として "--use-kubectl" を指定します。この場合、センサーはネイティブの kubernetes 機能を使用してイメージ スキャンを開始します。この引数を省略すると、イメージ コンテナは 'docker run' を使用して起動されます。

```
args: ["--k8s-mode", "--use-kubectl"]
```

TLS 認証が有効になっている場合は、args に Docker クライアント証明書、クライアント秘密鍵、および CA 証明書の名前を指定します。

```
args: ["--k8s-mode", "--tls-cacert","<file name of the CA
certificate that was used to sign docker server certificate>", "--
tls-cert", "<docker client certificate file name>", "--tls-key",
 "<docker client private key file name>"]
```

注: 3 つのファイルのいずれかにそれぞれ ca.pem、cert.pem、key.pem などのデフォルト名がある場合は、対応する引数を省略できます。

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「--mask-env-variable」パラメータを追加します。

```
args: ["--k8s-mode", "--mask-env-variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「--disableImageScan」パラメーターを追加します。

```
args: ["--k8s-mode", "--disableImageScan"]
```

スキャンポリシーを指定する場合は、args に "--scanning-policy" パラメーターを追加します。スキャン ポリシーで使用できる値は、"DynamicScanningOnly"、"StaticScanningOnly"、および "DynamicWithStaticScanningAsFallback" です。

```
args: ["--k8s-mode", "--scanning-policy", "StaticScanningOnly"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「--disable-log4j-scanning」パラメータを追加します。

```
args: ["--k8s-mode", "--disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "--disable-log4j-static-detection" パラメーターを追加します。

```
args: ["--k8s-mode", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「--optimize-image-scans」パラメーターを追加します。

```
args: ["--k8s-mode", "--optimize-image-scans"]
```

SCA スキャン機能がある場合は、args に "--perform-sca-scan" パラメーターを追加することで、コンテナイメージの SCA スキャンを有効にできます。

```
args: ["--k8s-mode", "--perform-sca-scan"]
```

デフォルトでは、SCA スキャンはオンライン・モードで実行されます。SCA スキャンのインターネット・アクセスを使用不可にして、スキャンをオフライン・モードで実行することもできます。注 - SCA スキャンはオンライン・モードで実行することをお勧めします。Java のソフトウェア・パッケージ列挙の品質は、SCA スキャンがオフライン・モードで実行されると大幅に低下します。正確なパッケージ検出のために、リモートの Maven リポジトリを参照する必要がある場合があります。これは、イメージの脆弱性ポスチャの精度に影響を与える可能性があります。

```
args: ["--k8s-mode", "--perform-sca-scan" "--disallow-internet-
access-for-sca"]
```

デフォルトの SCA スキャン・コマンド・タイムアウトは 5 分 (300 秒) です。デフォルトのタイムアウトは、秒単位で指定した新しい値で上書きできます。例えば、大きなコンテナ・イメージをスキャンするときには、SCA スキャンが完了する時間を確保するために、SCA スキャンのタイムアウトを増やす必要がある場合があります。

```
args: ["--k8s-mode", "--perform-sca-scan" "--sca-scan-timeout-in-
seconds=600"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-malware-
detection"]
```

[リソース] で 、次の項目を指定します。

```
resources: limits: cpu: "0.2" # Default CPU usage limit on
each node for sensor.
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
 limits:
     cpu: "0.2" # Default CPU usage limit on each node for sensor
      memory: "500Mi"
      requests:
       memory: "300Mi"
```

Container Sensor にメモリリソースの値(制限またはリクエスト)のいずれかが指定され、args に「--use-kubectl」が指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
    value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required) -
      name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required) -
      name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
  value: "30"
```

TLS 認証を有効にするには、次の 2 行のコメントを解除します。

```
- name: DOCKER_TLS_VERIFY
  value: "1"
```

注:TLS を無効にするには、DOCKER_TLS_VERIFY=""(空の文字列)を使用するか、削除するか、yml ファイルにコメントを付けたままにします。

注: TLS 経由で Docker デーモンとのセンサー通信を有効にすることで、お客様は docker 認証プラグインを使用して docker ソケットへのセンサーのアクセスを制限できます。

注:TLS 認証が有効で、DOCKER_HOST が指定されていない場合、センサーは実行されているワーカー・ノードの FQDN を自動的に検出し DOCKER_HOST センサー内の環境変数を<ワーカー・ノードの FQDN>:<2376>に設定します(2376 は Docker デーモンのデフォルトの TLS TCP ポートです)

注: 以下を追加して、自分で DOCKER_HOST を 127.0.0.1:<port#> または localhost:<port#> に設定します。

```
- name: DOCKER_HOST
     value: "<loopback IPv4 address or hostname>:<port#>"
```

注: DOCKER_HOST に設定されている FQDN、ホスト名、または IPv4 アドレスが、各ワーカー ノードの Docker サーバー証明書の CN またはサブジェクトの別名と一致していることを確認してください

TKGI セットアップでは、docker.sock は /var/run の場所で使用できません。ワーカー・ノードで docker.sock を見つけ、yaml ファイルでソケットとボリュームのマッピングを変更する必要があります。たとえば、docker.sock が /var/vcap/data/sys/run/docker/docker.sock にある場合は、ボリュームのソケットとボリュームのマッピングを次のように変更します。

```
volumes:
- name: socket-volume
  hostPath:
     path: /var/vcap/data/sys/run/docker/docker.sock
      type: Socket
```

TLS 認証を有効にする必要がある場合は、ボリュームの下の tls-cert-path のコメントを解除し 、クライアント証明書のディレクトリパスを指定するか、不要な場合はそのままにしておきます。

```
#- name: tls-cert-path
#   hostPath:
 #      path: <Path of directory of client certificates>
  #     type: Directory
```

TLS を使用している場合は、volumeMounts の下の部分のコメントも解除します。それ以外の場合は、コメントアウトしたままにします。

```
#- mountPath: /root/.docker
#name: tls-cert-path
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の **proxy-cert-path** のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
#   hostPath:
#       path: /root/cert/proxy-certificate.crt
#       type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、

任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
#   name: proxy-cert-path
```

cssensor-ds.yml **ファイルを変更したら**、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します。

```
kubectl create -f cssensor-ds.yml
```

Qualys Container Sensor をアンインストールする必要がある場合は、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

TKGI を使用した Kubernetes へのデプロイ - Containerd ランタイム

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

Kubernetes にデプロイする Qualys センサーの DaemonSet を作成するには、次の手順を実行します。

注: Container Sensor に、永続ストレージと Docker デーモン ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

cssensor-containerd-ds.yml ファイルを変更します

Qualys コンテナセンサーのテンプレートは https://github.com/Qualys/cs_sensor から直接 QualysContainerSensor.tar.xz を入手下さい。

重要: yml ファイルのフィールド配置は非常に重要です。テンプレートで提供されているフォーマットを必ず使用してください。

Qualys Container Sensor DaemonSet は、Kubernetes API サーバーと通信するための適切なアクセス許可を持つ ServiceAccount の一部として 'qualys' 名前空間にデプロイする必要があります。Role、ClusterRole、RoleBinding、および ClusterRoleBinding は、必要なアクセス許可を ServiceAccount に割り当てるために使用されます。「qualys」以外の別の名前空間で Qualys Container Sensor を既に実行している場合は、他の名前空間から Qualys Container Sensor をアンインストールし、「qualys」名前空間に新たにデプロイする必要があります。

次の権限が必要です。

get、list、watch - クラスター全体の脆弱性をスキャンするリソースを監視します。

作成、削除、削除コレクション - イメージの脆弱性評価用のコンテナを 'Qualys' 名前空間に生成するには、クラスター全体でポッドをスキャンし、その後クリーンアップします

注: コンテナ・センサーに、永続ストレージと containerd デーモン・ソケットへの読み取りおよび書き込みアクセス権があることを確認します。

yaml ファイルのパラメーターを変更する

cssensor-containerd-ds.yml ファイルを Kubernetes クラスターのマスター ノードにコピーし、次のパラメーターの値を指定して変更します。yaml ファイルが正しく機能するようにするには、以下で指定されているパラメーター/セクションのみを更新してください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

ここでは、一般的なセンサーを想定しています。spec の tolerations セクションのコメントを解除して、Sensor daemonset をマスターノードにデプロイします。

```
spec:
      #tolerations:
      # this toleration is to have the daemonset runnable on master
      nodes
      # remove it if want your masters to run sensor pod
      #- key: node-role.kubernetes.io/master
      # effect: NoSchedule

containers:
      - name: qualys-container-sensor
      image: <CS Sensor image name in the private/docker hub registry>
      args: ["--k8s-mode", "--container-runtime", "containerd"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--container-runtime", "containerd", "--
      registry-sensor"]
```

CICD センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--container-runtime", "containerd", "—cicd-
      deployed-sensor"]
```

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
      args: ["--k8s-mode", "--container-runtime", "containerd", "--
      loglevel",  "5"]
```

既定値 4 以外のスキャン スレッド値でセンサーを起動する場合は、args の追加値として "--scan-thread-pool-size"、"<number of threads>" を指定します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "—scan-thread-pool-size", "6"]
```

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルあたりのサイズを定義する場合は、args の追加値として "--log-filesize"、"<digit><K/M/>" ("K" はキロバイト、"M" はメガバイト)、および "--log-filepurgecount"、"<digit>" を指定します。既定値は "--log-filesize": "10M" と "--log-filepurgecount": "5" で、config を介して適用されます。

"--log-filesize": ログファイルあたりの最大サイズを定義するために使用できます。たとえば、"10K" (キロバイト)、"10M" (メガバイト)、"10" (バイト) などです。

"--log-filepurgecount":生成されるアーカイブログファイルの数を定義するために使用できますが、log/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--logfilesize", "5M", "--log-filepurgecount", "4"]
```

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、args に「--mask-env-variable」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "—mask-env-variable"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「--disableImageScan」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--disableImageScan"]
```

スキャンポリシーを指定する場合は、args に "--scanning-policy" パラメーターを追加します。スキャン ポリシーで使用できる値は、"DynamicScanningOnly"、"StaticScanningOnly"、および "DynamicWithStaticScanningAsFallback" です。

```
args: ["--k8s-mode", "--scanning-policy", "StaticScanningOnly"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「--disable-log4j-scanning」パラメータを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "--disable-log4j-static-detection" パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「--optimize-image-scans」パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--optimize-image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーとレジストリー・センサーでのみサポートされることに注意してください

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-malware-
detection"]
```

[リソース] で 、次の項目を指定します。

```
resources:
limits:
cpu: 0.2 # Default CPU usage limit (20% of one core on the host).
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
 limits:
    cpu: "0.2" # Default CPU usage limit on each node for sensor
   memory: "500Mi"
   requests:
   memory: "300Mi"
```

Container Sensor にメモリリソース値(制限または要求)のいずれかが指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
    value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required) -
      name: ACTIVATIONID
      value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required) -
      name: CUSTOMERID
      value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

docker hub image を使用する場合 POD_URL を指定します。それ以外の場合は、削除します。

```
- name: POD_URL
  value: <Specify POD URL>
```

スキャン コンテナの起動タイムアウトを分単位で指定します。この環境変数が存在しない場合は、10 分がデフォルトです。

```
- name: QUALYS_SCANNING_CONTAINER_LAUNCH_TIMEOUT
  value: "10"
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
- name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
  value: "30"
```

プロキシ情報のコメントを解除してインデントするか、不要な場合はそのままにしておきます。

- センサーとバックエンド (Container Management Service) 間の通信の場合:

```
#- name: qualys_https_proxy
#   value: <proxy FQDN or Ip address>:<port#>
```

- レジストリー・センサー・バージョン 1.21.0 以降がパブリック・レジストリーに使用される場合:

```
#- name: https_proxy
#  value: <proxy FQDN or Ip address>:<port#>
```

ボリュームの下の proxy-cert-path のコメントを解除するか、不要な場合はそのままにします。

```
#- name: proxy-cert-path
#  hostPath:
#      path: /root/cert/proxy-certificate.crt
#      type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。アクティベーション ID とカスタマー ID を取得するには、コンテナセキュリティ UI にログインし、[Configurations > Sensors] に移動して [Download] をクリックし、任意のセンサー タイプをクリックします。[インストール手順]画面のインストールコマンドには、アクティベーション ID とカスタマーID が含まれています。アクティベーション ID はパスワードのようなものなので、共有しないでください。

https プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分にコメントを付けたままにしている場合は、volumeMounts から次の部分もコメントしたままにしておくことができます 。

```
#- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
#   name: proxy-cert-path
```

Container Sensor DaemonSet をデプロイする方法

cssensor-containerd-ds.yml ファイルを変更したら 、Kubernetes マスターで次のコマンドを実行して DaemonSet を作成します。

```
kubectl create -f cssensor-containerd-ds.yml
```

Container Sensor DaemonSet を削除する方法

Qualys Container Sensor をアンインストールするには、Kubernetes マスターで次のコマンドを実行します。

```
kubectl delete -f cssensor-containerd-ds.yml
```

注: 永続ストレージは、各ワーカー・ノードで手動で削除する必要があります。

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、cssensor-containerd-ds.yml ファイルを変更し、args の追加値として "--sensor-without-persistent-storage" を指定します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
sensor-without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath: /usr/local/qualys/qpa/data
name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
 - name: persistent-volume  hostPath:
   path: /usr/local/qualys/sensor/data
       type: DirectoryOrCreate
```

CICD センサーのターゲット画像にタグを付ける方法

CICD イメージに qualys_scan_target:<tag> のタグを付けてスキャン対象としてマークするには、**nerdctl** ツールを使用します。画像に qualys_scan_target:<anytag> でタグを付けるには、<image-repo:tag> を使用します。

```
nerdctl -n k8s.io tag <image-repo:tag> qualys_scan_target:<anytag>
```

例えば

```
./nerdctl -n k8s.io tag docker.io/library/known:latest
  qualys_scan_target:known
```

メモ:

- NERDCTL バイナリは、ホスト上で使用可能である必要があります。
- ターゲット イメージは、K8S.IO 名前空間にのみ作成する必要があります。

Rancher を使用した Kubernetes へのデプロイ - Docker ランタイム

このセクションでは、センサー イメージがあることを前提としています。 コンテナ センサー イメージの取得

cssensor-ds.yml ファイルを変更する

cssensor-ds.yml ファイル (QualysContainerSensor.tar.xz から抽出) を変更して、次のパラメーターの値を指定します。yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

すべての Kubernetes ノードに、指定された URL からの最新の Qualys センサー イメージがあることを確認します。

```
containers:
      - name: qualys-container-sensor
      image: <CS Sensor image name in the docker hub/private registry>
      args: ["--k8s-mode"]
```

CI/CD 環境用にセンサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--cicd-deployed-sensor"]
```

レジストリー・センサーをデプロイする場合は、args 値を次のように指定します。

```
      args: ["--k8s-mode","--registry-sensor"]
```

General Sensor のイメージスキャンを無効にする場合は、args に「--disableImageScan」パラメーターを追加します。

```
args: ["--k8s-mode", "--disableImageScan"]
```

コンテナイメージで log4j 脆弱性スキャンを無効にする場合は、args に「--disable-log4j-scanning」パラメータを追加します。

```
args: ["--k8s-mode", "--disable-log4j-scanning"]
```

動的/静的イメージスキャンの log4j 静的検出を無効にする場合は、args に "--disable-log4j-static-detection" パラメーターを追加します。

```
args: ["--k8s-mode", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、args に「--optimize-image-scans」パラメーターを追加します。

```
args: ["--k8s-mode", "--optimize-image-scans"]
```

SCA スキャン機能がある場合は、args に "--perform-sca-scan" パラメーターを追加することで、コンテナイメージの SCA スキャンを有効にできます。 `args: ["--k8s-mode", "--perform-sca-scan"]`

デフォルトでは、SCA スキャンはオンライン・モードで実行されます。SCA スキャンのインターネット・アクセスを使用不可にして、スキャンをオフライン・モードで実行することもできます。注 - SCA スキャンはオンライン・モードで実行することをお勧めします。Java のソフトウェア・パッケージ列挙の品質は、SCA スキャンがオフライン・モードで実行されると大幅に低下します。正確なパッケージ検出のために、リモートの Maven リポジトリを参照する必要がある場合があります。これは、イメージの脆弱性ポスチャの精度に影響を与える可能性があります。

```
args: ["--k8s-mode", "--perform-sca-scan" "--disallow-internet-access-for-sca"]
```

デフォルトの SCA スキャン・コマンド・タイムアウトは 5 分 (300 秒) です。デフォルトのタイムアウトは、秒単位で指定した新しい値で上書きできます。例えば、大きなコンテナ・イメージをスキャンするときには、SCA スキャンが完了する時間を確保するために、SCA スキャンのタイムアウトを増やす必要がある場合があります。

注:「--sca-scan-timeout-in-seconds」パラメータは、シークレットとマルウェアの検出にも適用されます。

```
args: ["--k8s-mode", "--perform-sca-scan" "--sca-scan-timeout-inseconds=600"]
```

コンテナ イメージのシークレット検出を有効にする場合は、args に "--perform-secretdetection" パラメーターを追加します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、args に "--performmalware-detection" パラメーターを追加します。

```
args: ["--k8s-mode", "--registry-sensor", "--perform-secret-
detection"]
```

コンソールでログを出力する場合は、args の追加値として "--enable-console-logs" を指定します。

CPU 使用率を特定の値に制限するには、次のように変更します。 (オプション) 「リソース」で、 次の項目を指定します。

```
resources:
limits:
cpu: "0.2" # Default CPU usage limit(20% of one core on the host).
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。

たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

必要に応じて、Container Sensor のメモリ リソースを指定する場合は、resources で指定できます。Container Sensor のメモリ要求とメモリ制限の推奨値は次のとおりです。

```
resources:
 limits:
 cpu: "0.2" # Default CPU usage limit on each node for sensor
 memory: "500Mi"
requests:
 memory: "300Mi"
```

Container Sensor にメモリリソースの値(制限またはリクエスト)のいずれかが指定され、args に「--use-kubectl」が指定されている場合、メモリ要求とメモリ制限の両方がイメージスキャンコンテナに自動的に適用されます。デフォルト値は、それぞれ 200Mi と 700Mi です。

さらに、env で次の変数を指定することで、一方または両方の値を上書きできます。この例では、値を 300Mi と 800Mi に変更しました。

```
- name: QUALYS_SCANNING_CONTAINER_MEMORYREQUESTMB
  value: "300Mi"
- name: QUALYS_SCANNING_CONTAINER_MEMORYLIMITMB
  value: "800Mi"
```

env で 、次の項目を指定します。

```
Activation ID (Required)
```

```
- name: ACTIVATIONID
```

```
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Customer ID (Required)

      - name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

プロキシ情報を指定するか、不要な場合は削除します。

```
      - name: qualys_https_proxy
        value: proxy.localnet.com:3128
```

スキャンのたびに、ノードのステータスをチェックして、ノードがスケジュール可能かどうかを確認し、ノードがスケジュール可能な場合にのみスキャンを開始します。ノードのステータスがノードがスケジュール不可能であることを示している場合は、デフォルトの間隔である 15 分後にスキャンを再試行します。別のスキャン再試行間隔を分単位で指定することで、センサーがスキャンを再試行するまでに待機する時間を増減できます。

```
      - name: UNSCHEDULABLE_NODE_SCAN_RETRY_INTERVAL
        value: "30"
```

ボリュームでプロキシ証明書のパスを指定するか、不要な場合は削除します。

```
      - name: proxy-cert-path
        hostPath:
             path: /root/cert/proxy-certificate.crt
            type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。

プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

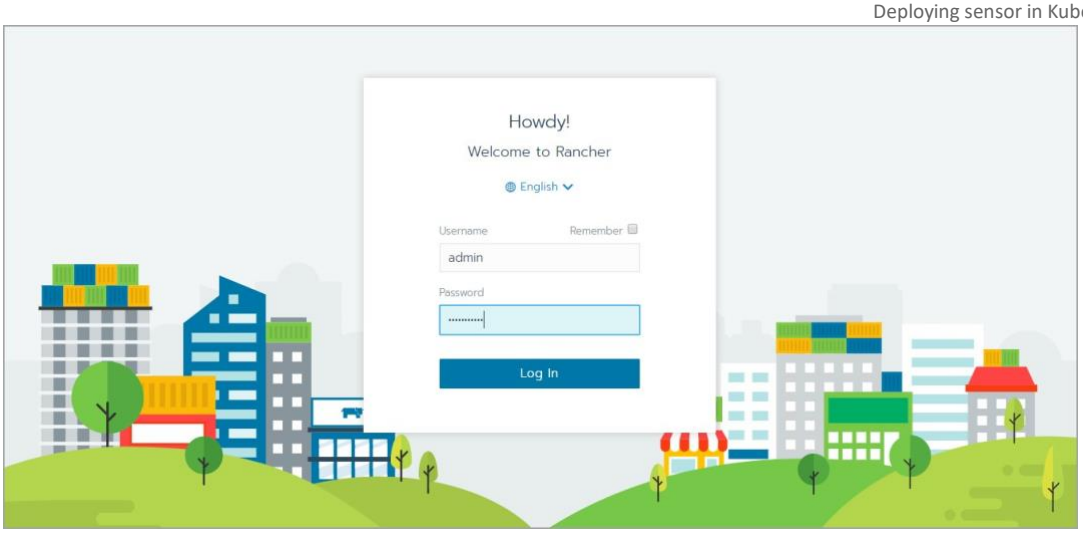
プロキシを使用しておらず、上記の部分を削除した場合は、volumeMounts から次の部分も削除できます 。

```
      - mountPath: /etc/qualys/qpa/cert/custom-ca.crt
        name: proxy-cert-path
```

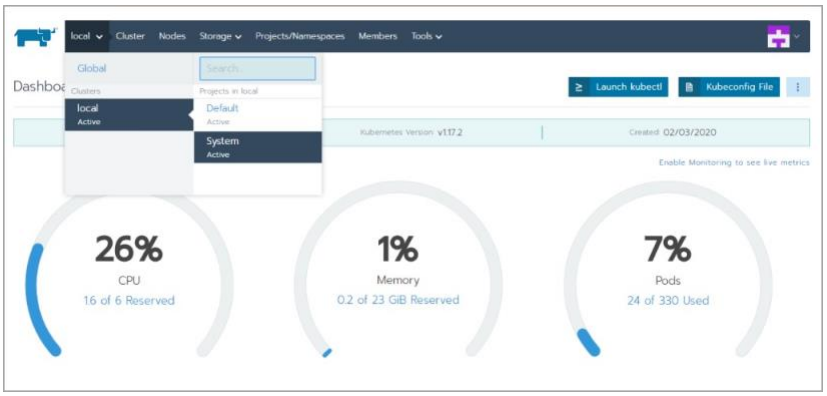
cssensor-ds.yml ファイルを変更したら 、保存します。

RancherUI での Qualys センサーDaemonSet の作成

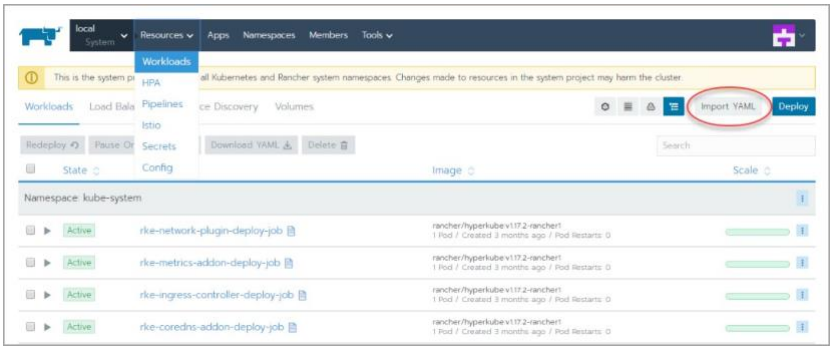
Rancher UI にログインして、Qualys センサーの DaemonSet を作成します。作成のセットアップ時に設定された資格情報を使用します。



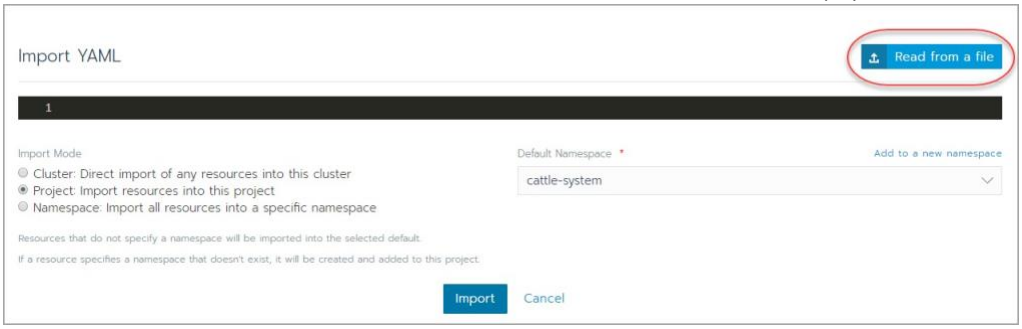
トップメニューから、Qualys センサーの DaemonSet を Rancher に展開するクラスタとプロジェクトを選択します。



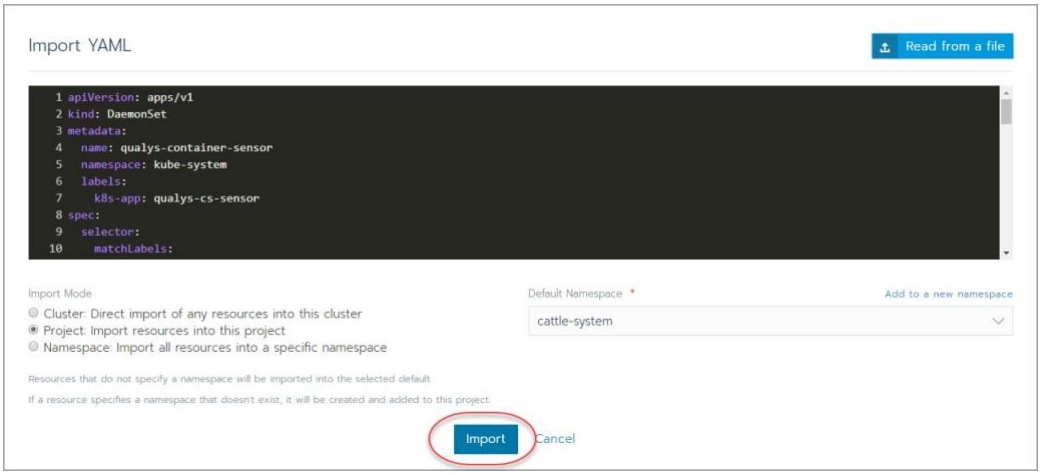
「リソース」タブに移動します。「YAML のインポート」ボタンをクリックします。



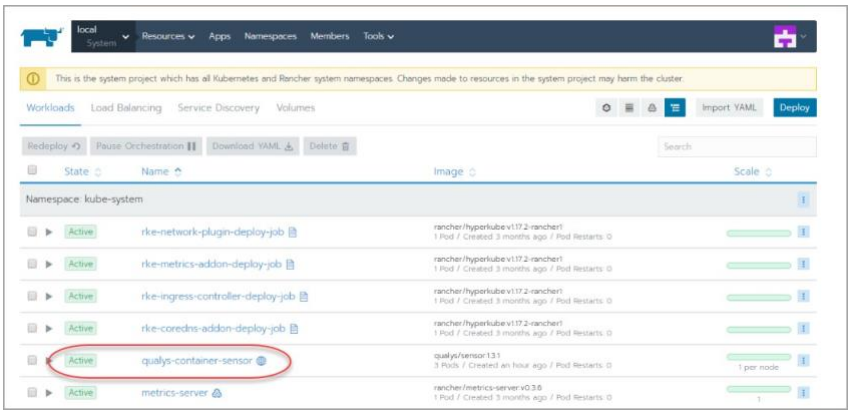
[ファイルから読み取る] ボタンをクリックし、変更した cssensor-ds.yml ファイルを参照して選択します。



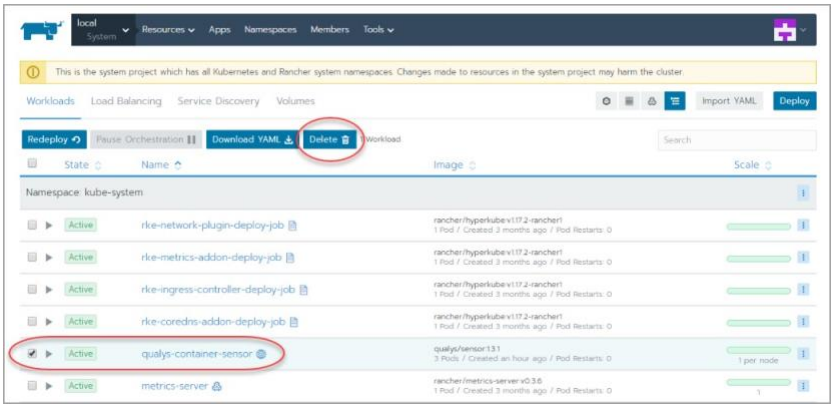
[インポート] ボタンをクリックします。



kube-system 名前空間の [ワークロード] ページで、qualys-containersensor DaemonSet がデプロイされ、アクティブになっていることを確認します。



Qualys Container Sensor をアンインストールする必要がある場合は、kube-system 名前空間で、qualys-container-sensor の横にあるチェックボックスをオンにして、[削除]をクリックします。



永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、 **cssensor-ds.yml** ファイルを変更し、args の追加値として "--sensor-without-persistent-storage" を指定します。

```
args: ["--k8s-mode","--sensor-without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

volumeMounts で 、persistent-volume セクションを削除/コメント化します。

```
volumeMounts:
 - mountPath: /usr/local/qualys/qpa/data
 name: persistent-volume
```

volumes で 、persistent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume
  hostPath:
       path: /usr/local/qualys/sensor/data
       type: DirectoryOrCreate
```

Google Kubernetes Engine(GKE)でマルチノード クラスタを使用してデプロイする

Google Kubernetes Engine(GKE)は永続ボリュームをクラスタ リソースとして扱い、クラスタのすべてのノードで共有されます。センサーの状態と構成を保持するために Persistent Volume Claim(PVC)を使用する現在の方法は、各センサーが同じ PVC ストレージの場所に書き込もうとするとセンサーに障害が発生するため、1 ノードを超える GKE クラスタでは機能しません。

センサーを GKE のマルチノード クラスタにデプロイするには、以下の手順をご覧ください。ランタイムに応じて正しい yaml ファイルを選択してください。Docker ランタイムの場合は、**cssensor-ds.yml** を使用します(GKE ノード バージョン 1.18 以前に関する手順をご覧ください)。Containerd ランタイムの場合は、**cssensorcontainerd-ds.yml** を使用します(GKE ノード バージョン 1.19 以降の手順をご覧ください)。

GKE ノード バージョン 1.18 以前

センサーを GKE のマルチノード クラスタにデプロイするには、以下の手順に従います。このセクションの手順は、バージョン 1.18 以前の GKE ノードに適用されます。

cssensor-ds.yml を次の引数で変更します。

```
args: ["--k8s-mode", "--sensor-without-persistent-storage", "--
enable-console-logs"]
```

volumeMounts で 、persistent-volume セクションと agent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath: /usr/local/qualys/qpa/data
name: persistent-volume
- mountPath: /usr/local/qualys/qpa/data/conf/agent-data
name: agent-volume
```

ボリュームで 、persistent-volume セクションと agent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume
hostPath:
   path: /usr/local/qualys/sensor/data
   type: DirectoryOrCreate
- name: agent-volume
 hostPath:
   path: /etc/qualys
   type: DirectoryOrCreate
```

GKE ノード バージョン 1.19 以降

センサーを GKE のマルチノード クラスタにデプロイするには、以下の手順に従います。このセクションの手順は、GKE ノード バージョン 1.19 以降に適用されます。

永続ストレージを使用しないデプロイの手順

cssensor-containerd-ds.yml を次の引数で変更します。

```
args: ["--k8s-mode", "--container-runtime", "containerd", "--
sensor-without-persistent-storage", "--enable-console-logs"]
```

volumeMounts で 、persistent-volume セクションと agent-volume セクションを削除/コメント化します。

```
volumeMounts:
- mountPath: /usr/local/qualys/qpa/data  name: persistent-volume
- mountPath: /usr/local/qualys/qpa/data/conf/agent-data  name:
agent-volume
```

ボリュームで 、persistent-volume セクションと agent-volume セクションを削除/コメント化します。

```
volumes:
- name: persistent-volume
    hostPath:
   path: /usr/local/qualys/sensor/data
    type: DirectoryOrCreate
- name: agent-volume
  hostPath:
        path: /etc/qualys
        type: DirectoryOrCreate
```

永続ストレージを使用してデプロイする手順

「Kubernetes のデプロイ - Containerd ランタイム」セクションで説明されているデプロイ手順に従い、ボリュームの下の永続ストレージ パスに次の変更を加えます。

デフォルトのパスは GKE ノードで書き込み可能ではないため、[volumes] で永続ストレージ パスを「/usr/local/qualys/sensor/data」から「/tmp/qualys/sensor/data」に変更します。

```
volumes:
- name: persistent-volume
 hostPath:
  path: /tmp/qualys/sensor/data
  type: DirectoryOrCreate
```

Helm チャートを使用した Kubernetes へのデプロイ

Helm は Kubernetes のパッケージ マネージャーであり、Helm チャートを使用して Kubernetes に Container Security Sensor をデプロイできます。

始める前に:

- 次のアプリケーションバージョンがあることを確認します。

  Kubernetes:

  - Kubernetes 1.17+

  - Helm v3.x.x

  RedHat OpenShift:

  - OpenShift v4.0+ (The default container engine is CRI-O.)

- 次のコマンドを実行して、コンテナ ランタイムの詳細を取得します。

```
kubectl get nodes -o wide
```

Qualys コンテナセキュリティセンサーバージョンの Helm チャートのサポート

Sensor Version	Helm Chart Version
1.19.2-0	1.0.0

Helm チャートのインストール

1 Qualys Helm チャートは、次の場所からダウンロードします。

'https://github.com/Qualys/Qualys-Helm-Charts' or
'https://artifacthub.io/packages/helm/??'

2 Install the Helm chart using the following command:

```
helm install [NAME] [CHART] [flags]
```

For example,

```
helm install qcs-sensor-demo qcs-sensor --namespace qualys --
create-namespace
```

3 Qualys Helm チャートのパラメータを設定します。'helm install' の '--set key=value[,key=value]' 引数を使用して各パラメーターを指定します。

例えば

```
helm install qcs-sensor-demo qcs-sensor --namespace qualys --
set containerd.enabled=true
```

注: コマンドラインパラメータ値は、values.yaml のパラメータ値よりも優先されます。

Helm チャートの構成パラメーター

以下は、Helm チャートの構成可能なパラメーターです。

Parameter	Mandatory?	Description	Default Value
containerd.enabled	Enable only one runtime environment.	Set to true, if the container runtime is containerd.	true
containerd.socketPath	Optional	The path of the mounted volume for the containerd socket.	/var/run/containerd/containerd.sock
crio.enabled	Enable only one runtime environment.	Set to true, if the container runtime is CRI-O.	false
crio.socketPath	Optional	The path of the mounted volume for the CRI-O socket.	/var/run/crio/crio.sock
docker.enabled	Enable only one runtime environment.	Set to true, if the container runtime is docker.	false
docker.socketPath	Optional	The path of the mounted volume for the docker socket.	/var/run/docker.sock
docker.tlsVerify.enabled	Optional	Enables the TLS authentication. The value should be 0 or 1.	false

Parameter	Mandatory?	Description	Default Value
docker.tlsVerify.tlsCertPath	Optional (Mandatory if DOCKER_TLS_VERIFY=1 is defined.)	Provide the path of the client certificate directory.	-
docker.tlsVerify.dockerHost	Optional (Mandatory if DOCKER_TLS_VERIFY=1 is defined.)	Specify the address on which the docker daemon is configured to listen.	-
docker.tlsVerify.dockerHostValue	Optional	Specify the loopback IPv4 address or hostname, and port <IPv4 address or hostname>:<port#>.	-
openshift		Set to true, if deploying in OpenShift.	false
qualys.createNamespace	Optional	Set to true, if you want to create a new custom namespace.	false
qualys.namespace	Optional	Provide the namespace to be used. Use the same namespace in values.yaml and on command line when using the helm install command.	qualys
qualys.customerID	Mandatory	Provide the Qualys customer id.	-
qualys.activationID	Mandatory	Provide the Qualys activation id.	-
qualys.pod_url	Mandatory	Provide the URL of a Qualys POD.	-
qualys.containerLaunchTimeout	Optional	Speicfy the launch timeout for the scanning container in minutes.	10
qualys.image	Optional	Specify the name of the Container Security sensor image in the private/dockerhub registry.	qualys/qcs-sensor:1.16.0-0
qualys.imagePullPolicy	Optional	Specify how to pull (download) the specified image.	IfNotPresent
qualys.cpu	Optional	Specify the CPU usage limit in percentage for the sensor. Range: 0-100.	0.2 (20% per core on the host)
qualys.args.withoutPersistentStorage	Optional	Runs the sensor without using the persistent storage on the host.	false

Parameter	Mandatory?	Description	Default Value
qualys.args.enableConsoleLogs	Optional	Prints logs on the console.	false
qualys.args.cicdDeployedSensor	Optional	Runs the sensor in a CI/CD environment.	false
qualys.args.registrySensor	Optional	Runs the sensor to list and scan the registry assets.	false
qualys.args.concurrentScan	Optional	Specify the number of docker/registry asset scans to run in parallel. Range: 1-20.	4
qualys.args.disableLog4jScanning	Optional	Disables the log4j vulnerability scanning for container images.	false
qualys.args.disableLog4jStaticDetection	Optional	Disables the log4j static detection for dynamic/static image scans.	false
qualys.args.logFilePurgeCount	Optional	The maximum number of sensor log files to archive.	5
qualys.args.logFileSize	Optional	The maximum size for a sensor log file in bytes. You can specify "<digit><K/M/>", where K is kilobytes and M is megabytes.	10M
qualys.args.logLevel	Optional	Sets the logging level for sensor. It determines the type of sensor data you want to log. Specify a value from 0 to 5. 0= Error, 1= Warning, 3= Information, 4= Verbose, 5= Trace	3 (Information)
qualys.args.maskEnvVariable	Optional	Masks the environment variables for images and containers.	false
qualys.args.optimizeImageScans	Optional	Optimizes the image scans for the General sensor. It is available for the General sensor type only.	false
qualys.args.disableImageScan	Optional	Disables the image scans for the General sensor.	false
qualys.readOnly	Optional	Runs the sensor in read-only mode.	-
qualys.tolerations.enabled	Optional	Allows the DaemonSet runnable on master nodes.	false
qualys.tolerations.toleration.key	Optional	Specify the toleration key.	-

Parameter	Mandatory?	Description	Default Value
qualys.tolerations.toleration.operator	Optional	Specify the toleration operator.	-
qualys.tolerations.toleration.value	Optional	Specify the toleration value.	-
qualys.tolerations.toleration.effect	Optional	Specify the toleration effect.	-
qualys.proxy.enabled	Optional	Set to true, if a proxy is required to connect to the Qualys cloud.	false
qualys.proxy.proxyvalue	Optional	Specify the IPv4/IPv6 address or FQDN of the proxy server.	-
qualys.proxy.proxycertpath	Optional	Specify the path of the proxy certificate file. proxycertpath is applicable only if the proxy has a valid certificate file.	-
qualys.sensorContResources.enabled	Optional	Specifies the memory resources for the Sensor container.	false
qualys.sensorContResources.memoryLimit	Optional	Specify the memory usage limit for the sensor container.	-
qualys.sensorContResources.memoryRequest	Optional	Specify the memory usage request for the sensor container.	-
qualys.scanningContResources.enabled	Optional	Specifies the memory resources for the scanning container.	false
qualys.scanningContResources.memoryLimit	Optional	Specify the memory usage limit for the scanning container.	-
qualys.scanningContResources.memoryRequest	Optional	Specify the memory usage request for the scanning container.	-
qualys.persistentvolhostpath	Optional	Specify the directory where the sensor will store the files.	/usr/local/qualys/sensor/data
qualys.persistentVolumeClaim.enabled	Optional	Requests for a storage of a specific size from the gross persistent volume.	false

Parameter	Mandatory?	Description	Default Value
qualys.persistentVolumeClaim.storageClassName	Mandatory if `qualys.persistentVolumeClaim.enabled` is `true`.	Specify the storage class name used by Kubernetes PersistentVolume.	-
qualys.persistentVolumeClaim.storage	Mandatory if `qualys.persistentVolumeClaim.enabled` is `true`.	Specify the storage memory required. For example, 1Gi for the general/cicd sensor and 10Gi for the registry sensor.	-
qualys.priorityClass.enabled	Optional	Set to true, if you want to use the priority and preemption on PODs.	false
qualys.priorityClass.priorityClassName	Optional	Specify the priority class name used in DaemonSet.	-
qualys.priorityClass.priorityClassValue	Optional	Specify the value that determines the priority. For example, "1000000". Enter an integer less than or equal to 1 billion (1000000000). The higher the value, the higher the priority. Values are relative to the values of other priority classes in the cluster. Reserve very high numbers for system critical pods that you don't want to be preempted (removed).	-
qualys.priorityClass.preemptionPolicy	Optional	Specify the preemption policy for a POD.	-
qualys.priorityClass.globalDefault	Optional	Indicates that the value of this PriorityClass should be used for PODs without a priorityClassName.	-

名前空間の使用法

- 新しいカスタム名前空間を作成するには、次のコマンドを使用します。

```
helm install qcs-sensor-demo qcs-sensor --set
qualys.createNamespace=true --set
qualys.namespace=namespace_name -n namespace_name –create-
namespace
```

- 既存の名前空間を作成するには、次のコマンドを使用します。

```
helm install qcs-sensor-demo qcs-sensor --set
qualys.namespace=existing_namespace_name -n
 existing_namespace_name
```

- デフォルト(Qualys)名前空間を使用するには、次のコマンドを使用します

```
helm install qcs-sensor-demo qcs-sensor -n qualys
```

Helm チャートのアップグレード

次のコマンドを使用して、チャートをアップグレードします。

```
helm upgrade [RELEASE] [CHART] [flags]
```

ここで、[RELEASE]はリリース名、[CHART]はチャートパスです。

例えば

```
helm upgrade qcs-sensor-demo qcs-sensor --namespace qualys
```

Helm チャートのアンインストール

次のコマンドを使用して、チャートをアンインストールします。

```
helm uninstall RELEASE_NAME [...] [flags]
```

例えば

```
helm uninstall qcs-sensor-demo --namespace qualys
```

Kubernetes クラスタ属性の収集

Sensor バージョン 1.8 および Container Security バージョン 1.10 以降、Kubernetes クラスタ属性のコレクションが追加されました。Container Security UI でコンテナまたはセンサーを検索するときに、センサーによって収集された Kubernetes クラスタ属性を検索できます。Kubernetes クラスターの属性には、ノードの詳細、ポッドの詳細、コントローラーの詳細などが含まれます。Container Security API を使用して、コンテナとセンサー用に収集された Kubernetes クラスタ属性を確認します。

重要 - Kubernetes 属性は、Container Security バージョン 1.10 のリリース以降に検出されたコンテナに対してのみ処理されます。Kubernetes 属性は、コンテナが初めて検出されたときに、コンテナ検査処理の一部として収集されます。Kubernetes の既存のデプロイの Kubernetes クラスター属性をフェッチするには、既存のデプロイを "ロールアウト再起動" して新しいコンテナを作成し、コンテナ検査処理を開始する必要があります。Kubernetes 属性は、Kubernetes クラスター上に新しく作成されたコンテナに対して収集されます。

次のコマンドを使用して「ロールアウト再起動」を行います。

```
kubectl rollout restart deployment <deployment-name> -n <namespace>
```

Kubernetes クラスターの属性は何ですか?

- Cluster type (Kubernetes)
- Cluster version
- Project name (collected for projects in Google Cloud Platform)

- Node name and flag indicating whether the node is the master node
- Pod name
- Pod UUID
- Pod namespace
- Pod labels (key and value pairs)
- Controller name
- Controller UUID
- Controller type (e.g. DaemonSet, Deployment, ReplicaSet, etc)

Kubernetes にデプロイされたセンサーを更新する

Kubernetes で Container Sensor DaemonSet を最新バージョンに更新できます。この情報は、Amazon Elastic Container Service for Kubernetes (Amazon EKS)、Google Kubernetes Engine (GKE)、および Azure Kubernetes Service (AKS) に適用されます。

Container Sensor に、永続ストレージと Docker デーモンソケットへの読み取りおよび書き込みアクセス権があることを確認します。

Kubernetes マスターで次の手順を実行して、Container Sensor を更新します。

注: Container Sensor DaemonSet が Kubernetes 環境で実行されていることを確認します。

注: 「qualys」以外の名前空間で Qualys コンテナ センサーをすでに実行している場合は、最初に他の名前空間からセンサーをアンインストールする必要があります。最新の QualysContainerSensor.tar.xz から抽出された新しい yml を使用するか、https://github.com/Qualys/cs_sensor から直接 yml ファイルをダウンロードできます。新しい Qualys Container Sensor を 'qualys' 名前空間にデプロイします。永続ストレージには、persistent-volume の hostpath で以前のデプロイと同じパスを使用する必要があります。

```
- name: persistent-volume
hostPath:
 path: /usr/local/qualys/sensor/data
  type: DirectoryOrCreate
```

Kubernetes マスターの Qualys クラウド ポータルから QualysContainerSensor.tar.xz ファイルをダウンロードします。

センサーパッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

バージョン情報ファイル (QualysContainerSensor.tar.xz から抽出) からセンサーのバージョンをコピーします。

cssensor-ds.yml ファイルを変更する

cssensor-ds.yml ファイル (QualysContainerSensor.tar.xz から抽出) を変更して、次のパラメーターの値を指定します。yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

すべての Kubernetes ノードに、指定された URL からの最新の Qualys センサー イメージがあることを確認します。

```
containers:
   - name: qualys-container-sensor
   image: <CS Sensor image name in the docker hub/private registry>
   args: ["--k8s-mode"]
```

画像の値は、次の形式にする必要があります。

```
registryurl/qualys/sensor:<version-info>
```

CI/CD 環境用にセンサーをデプロイする場合は、**args** 値を次のように指定します。

```
args: ["--k8s-mode","--cicd-deployed-sensor","--log-level","5","--
log-filesize","5M","--log-filepurgecount","4"]
```

レジストリー・センサーをデプロイする場合は、**args** 値を次のように指定します。

```
args: ["--k8s-mode","--registry-sensor","--log-level","5","--
logfilesize","5M","--log-filepurgecount","4"]
```

注: 上記の args 値の "--log-level"、"--log-filesize"、および "--log-filepurgecount" の値は サンプルにすぎません。必要に応じて適切な値を指定します。

コンソールでログを出力する場合は、`args` の追加値として "--**enable-console-logs**" を指定します。

CPU 使用率を特定の値に制限するには、次のように変更します。 (オプション)「リソース」で、 次の項目を指定します。

```
resources:
 limits:
  cpu: "0.2" # Default CPU usage limit(20% of one core on the host).
```

たとえば、CPU 使用率を 5% に制限するには、resources:limits:cpu: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、resources:limits:cpu 値を設定すると、CPU 制限が 1 つのコアのみに適用されます。

たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、resources:limits:cpu 値を 0 に設定します。

env で 、次の項目を指定します。

アクティベーション ID (必須: アップグレードする既存の Container Sensor DaemonSet で提供されているものと同じアクティベーション ID を使用します)

```
- name: ACTIVATIONID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

顧客 ID (必須: アップグレードする既存の Container Sensor DaemonSet で提供されているものと同じ顧客 ID を使用します)

```
- name: CUSTOMERID
        value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

プロキシ情報を指定するか、不要な場合は削除します。

```
- name: qualys_https_proxy
  value: proxy.localnet.com:3128
```

ボリュームでプロキシ証明書のパスを指定するか、不要な場合は削除します。

```
- name: proxy-cert-path
  hostPath:
  path: /root/cert/proxy-certificate.crt
  type: File
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。

プロキシを使用している場合は、センサーがコンテナ管理サーバと通信するための有効な証明書ファイルがすべての Kubernetes ノードにあることを確認します。

プロキシを使用しておらず、上記の部分を削除した場合は、volumeMounts から次の部分も削除できます 。

```
- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
  name: proxy-cert-path
```

cssensor-ds.yml を変更したら、ファイルを保存し、更新スクリプト (k8s-rollingupdate.sh) を実行する前に、Kubernetes マスターのレジストリに docker login を実行します。

例えば :

```
docker login mycloudregistry.com
```

レジストリは、すべての Kubernetes ノードと、更新が実行されている Kubernetes マスターからアクセスできる必要があります。

Container Sensor DaemonSet を最新バージョンに更新するには、Kubernetes マスターで次のコマンドを実行します。

```
./k8s-rolling-update.sh Registry_Url=mycloudregistry.com
```

注:k8s-rolling-update.sh は、docker load、docker tag、docker push をレジストリに実行します。

Docker Swarm でのセンサーのデプロイ

他のアプリケーション コンテナと同様に Container Sensor を DaemonSet に統合し、レプリケーション係数を 1 に設定して、Docker ホストに常にセンサーがデプロイされるようにします。

Docker Swarm にデプロイする Qualys センサーの DaemonSet を作成するには、次の手順を実行します。

Linux コンピュータ上の QualysContainerSensor.tar.xz ファイルを Qualys クラウド ポータルからダウンロードします。

センサーパッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

次のコマンドを使用して、Docker Swarm クラスター内のすべてのノードに共通するリポジトリに qualys センサー イメージをプッシュします。

```
sudo docker load -i qualys-sensor.tar
sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
sudo docker push <URL to push image to the repository>
```

例えば：

```
sudo docker load -i qualys-sensor.tar
sudo docker tag c3fa63a818df myregistry.com/qualys_sensor:xxx
sudo docker push myregistry.com/qualys_sensor:xxx
```

注: 例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

cssensor-swarm-ds.yml ファイルを変更する

cssensor-swarm-ds.yml ファイル (QualysContainerSensor.tar.xz から抽出) を変更して、次のパラメーターの値を指定します。yml ファイルを正しく機能させるには、以下で説明する各セクションを削除/コメント化しないようにしてください。yml ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

すべてのマスターノードとワーカーノードに、指定された URL からの最新の Qualys センサー イメージがあることを確認します。

```
 qualys-container-sensor:
     image: <CS Sensor image name in the docker hub/private registry>
      deploy:
            mode: global  # Deploy 1 container on each node == DaemonSet
         command: ["--swrm-mode"]
```

CI/CD 環境用にセンサーをデプロイする場合は、**コマンド**値を次のように指定します。

```
command: ["--swrm-mode","--cicd-deployed-sensor","—log-
level","5","--log-filesize","5M","--log-filepurgecount","4"]
```

Registry Sensor をデプロイする場合は、**コマンド**値を次のように指定します。

```
command: ["--swrm-mode","--registry-sensor","--log-level","5","-
log-filesize","5M","--log-filepurgecount","4"]
```

注:上記のコマンド値の「--log-level」、「--log-filesize」、および「--log-filepurgecount」の値は サンプルにすぎません。必要に応じて適切な値を指定します。

コンソールにログを出力する場合は、command の追加値として "--enable-console-logs" を指定します。

センサーログとコンテナセキュリティ UI でイメージとコンテナの環境変数をマスクする場合は、コマンドに「--mask-env-variable」を追加値として追加します。

CPU 使用率を特定の値に制限するには、次のように変更します。 (オプション) **[deploy]** で、次の項目を指定します。

```
mode: global    # Deploy 1 container on each node == DaemonSet
 resources:
  limits:
   cpus: '0.20' # Default CPU usage limit(20% of one core on the host.
```

たとえば、CPU 使用率を 5% に制限するには、deploy:resources:limits:cpus: "0.05" を設定します。これにより、CPU 使用率がホスト上の 1 つのコアの 5% に制限されます。

ノードに複数のプロセッサがある場合、deploy:resources:limits:cpus 値を設定すると、CPU 制限が 1 つのコアにのみ適用されます。

たとえば、システムに 4 つの CPU があり、CPU 制限を全体の CPU 容量の 20% に設定する場合、CPU 制限を 0.8 に設定する必要があります (つまり、1 つのコアのみの 80%) は、合計 CPU 容量の 20% になります。

CPU 使用率の制限を無効にするには、deploy:resources:limits:cpus の値を 0 に設定します。

[環境] で 、次の項目を指定します。

```
environment:
  ACTIVATIONID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
   CUSTOMERID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
   qualys_https_proxy: proxy.qualys.com:3128
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。不要な場合は、プロキシ情報を削除できます。

[ボリューム] で 、次の情報を入力していることを確認します。

```
volumes:
 - type: bind
```

```
      source: /var/run/
      target: /var/run/
    - type: volume
       source: persistent-volume
       target: /usr/local/qualys/qpa/data/
  - type: bind
      source: /etc/qualys  # Must exist !
      target: /usr/local/qualys/qpa/data/conf/agent-data
```

source は "persistent-volume" のままにします。これにより、ボリュームマッピングのソースディレクトリが docker swarm ルートディレクトリ (つまり、/data/docker/volumes) に設定されます。

ボリューム・マッピングを正常に行うには、すべてのマスター・ノードとワーカー・ノードに /etc/qualys ディレクトリーが存在している必要があります。

```
  volumes:
   persistent-volume:
```

[**configs**] で、次の情報を入力してください。

```
  configs:   proxy-cert-path:
    file: /root/cert/proxy-certificate.crt
```

プロキシを使用している場合は、すべてのマスター ノードとワーカー ノードに、センサーがコンテナ管理サーバーと通信するための有効な証明書ファイルがあることを確認します。

プロキシを使用しておらず、環境から qualys_https_proxy を削除した場合は、次の部分も削除できます。

```
configs:
  - source: proxy-cert-path
     target: /etc/qualys/qpa/cert/custom-ca.crt

    configs:   proxy-cert-path:
        file: /root/cert/proxy-certificate.crt
```

cssensor-swarm-ds.yml **ファイルを変更したら** 、docker swarm master/leader で次のコマンドを実行してスタックを作成します。

```
docker stack deploy -c cssensor-swarm-ds.yml qualys-container-sensor
```

Qualys Container Sensor をアンインストールする必要がある場合は、docker swarm master/leader で次のコマンドを実行します。

```
docker stack rm qualys-container-sensor
```

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、**cssensor-swarm-ds.yml** ファイルを変更し、command の追加値として "--sensor-without-persistent-storage" を指定します。

```
command: ["--swrm-mode","--sensor-without-persistent-storage"]
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

[**volumes** (outside **services**)] で、persistent-volume セクションを削除/コメント化します。

```
volumes:
 persistent-volume:
```

ボリューム(サービス内)で、persistent-volume セクションを削除/コメント化します。

```
services:
 volumes:
 - type: volume
   source: persistent-volume
   target: /usr/local/qualys/qpa/data/
```

AWS ECS クラスターでのセンサーのデプロイ

Qualys Container Sensor をデーモンサービスとして Amazon ECS クラスターにデプロイするには、次の手順を実行します。

前提条件: AWS ECS クラスターが稼働している必要があります。

Linux コンピュータ上の QualysContainerSensor.tar.xz **ファイルを** Qualys **クラウド ポータルから**ダウンロードします。

センサーパッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

次のコマンドを使用して、クラスタ内のすべてのノードに共通のリポジトリに qualys センサーイメージをプッシュします。

```
sudo docker load -i qualys-sensor.tar
sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
sudo docker push <URL to push image to the repository>
```

例えば：

```
sudo docker load -i qualys-sensor.tar
sudo docker tag c3fa63a818df 20576712438.dr.ecr.us-
east1.amazonaws.com/container-sensor:qualys-sensor-xxx
sudo docker push 20576712438.dr.ecr.us-east-1.amazonaws.com/container-
sensor:qualys-sensor-xxx
```

注: 例をそのまま使用しないでください。レジストリ/イメージのパスを独自のパスに置き換えます。

cssensor-aws-ecs.json ファイルを変更する

cssensor-aws-ecs.json ファイル (QualysContainerSensor.tar.xz から抽出) を変更して、次のパラメータの値を指定します。json ファイルを正しく機能させるには、以下の各セクションを削除/コメント化しないようにしてください。json ファイルは https://github.com/Qualys/cs_sensor から直接ダウンロードできることに注意してください

```
 "containerDefinitions": [
    {
       "name": "qualys-container-sensor",
       "image": "20576712438.dr.ecr.us-east-1.amazonaws.com/container
                 sensor:qualys-sensor-xxx",
       "cpu": 10,
       "memory": 512,
       "essential": true,
       "command": [
```

```
        "--ecs-mode"
        ],
```

cpu (vCPU の数) とメモリ (サイズ (MB 単位) に適切な値を指定します。

CI/CD 環境用にセンサーをデプロイする場合は、コマンド値を次のように指定します。

```
    "command": [
        "--ecs-mode",
        "--cicd-deployed-sensor",
        ],
```

Registry Sensor をデプロイする場合は、コマンド値を次のように指定します。

```
    "command": [
        "--ecs-mode",
        "--registry-sensor",
        ],
```

ログレベルを変更する場合は、args の追加値として「--log-level」、「<0〜5>の数値」を指定します。

```
    "command": [ "--ecs-mode", "--log-level", "5",]
```

生成されるアーカイブされた qpa.log ファイルの数とログ ファイルごとのサイズを定義する場合は、args に "--log-filesize"、"<digit></K/M>" ("K" はキロバイト、"M" はメガバイト)、および "log-filepurgecount"、"<digit>" を追加値として指定 **します**。デフォルトは "log-filesize": "10M" と "log-filepurgecount": "5" で、config で適用されます。

"--log-filesize": ログファイルあたりの最大サイズを定義するために使用できます。たとえば、"10K" (キロバイト)、"10M" (メガバイト)、"10" (バイト) などです。

"--log-filepurgecount": 生成するアーカイブ・ログ・ファイルの数を定義するために使用できます。ログ/ディレクトリには常に現在の qpa.log ファイルがあることに注意してください。

```
    "command": [ "--ecs-mode", "--log-level", "5", "--log-filesize",
    "5M", "--log-filepurgecount", "4"]
```

コンソールにログを出力する場合は、コマンドの追加値として「--enable-console-logs」を指定します。

General Sensor(すべてのランタイムでサポート)のイメージスキャンを無効にする場合は、「--disableImageScan」パラメーターを追加します。

```
    "command": [ "--ecs-mode", "--disableImageScan"]
```

「--disable-log4j-scanning」を使用して、コンテナイメージの Log4J 脆弱性スキャンを無効にすることができます。

```
    "command": [ "--ecs-mode", "--disable-log4j-scanning"]
```

「--disable-log4j-static-detection」を使用すると、動的/静的イメージスキャンの log4j 静的検出を無効にすることができます。

```
"command": [ "--ecs-mode", "--disable-log4j-static-detection"]
```

General Sensor のイメージスキャンを最適化する場合は、「--optimize-image-scans」パラメーターを追加します。

```
"command": [ "--ecs-mode", "--optimize-image-scans"]
```

コンテナ イメージのシークレット検出を有効にする場合は、"--perform-secretdetection" パラメーターを追加します。シークレット検出は、CICD センサーおよびレジストリー・センサーでのみサポートされることに注意してください。

```
"command": [ "--ecs-mode", "--perform-secret-detection"]
```

コンテナ イメージのマルウェア検出を有効にする場合は、"--performmalware-detection" パラメーターを追加します。マルウェア検出は、レジストリ センサーでのみサポートされていることに注意してください。

```
"command": [ "--ecs-mode", "--perform-malware-detection"]
```

[環境] で 、次の項目を指定します。

```
"environment": [
        {
          "name": "ACTIVATIONID",
          "value": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
        },
        {
          "name": "CUSTOMERID",
          "value": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
        },
        {
          "name": "POD_URL",
          "value": "<Specify POD URL>"
        },
        {
          "name": "qualys_https_proxy",
          "value": "<proxy FQDN or IP address>:<port#>"
        }
    ]
```

アクティベーション ID とカスタマー ID は必須です。サブスクリプションのアクティベーション ID と顧客 ID を使用します。プロキシ情報を指定するか、不要な場合はセクションを削除します。プロキシ セクションを削除する場合は、json のインデントが正しいことを確認してください。

プロキシを使用しておらず、環境から qualys_https_proxy を削除した場合は、mountPoint とボリュームから次の部分を削除できます。

```
configs:
 - source: proxy-cert-path
target: /etc/qualys/qpa/cert/custom-ca.crt

configs:
proxy-cert-path:
file: /root/cert/proxy-certificate.crt
```

proxy セクションが環境から削除されている場合は、**mountPoints** と **volumes** の下の **proxy-cert-path** セクションも削除します。

```
"mountPoints": [
            {
            "sourceVolume": "proxy-cert-path",
            "containerPath": "/etc/qualys/qpa/cert/custom-ca.crt"
            },
          ]

"volumes": [
        {
            "name": "proxy-cert-path",
            "host": {
                "sourcePath": "/root/cert/proxy-certificate.crt"
            }
            }
        ]
```

[ボリューム] で、persistent_volume の情報を入力します。persistent_volume のカスタムの場所を指定すると、Docker ホストでまだ使用できない場合は作成されます。変更が完了したら、**cssensor-aws-ecs.json** ファイルを保存します。

json ファイルを Amazon ECS UI にインポートして、セニョールのデプロイを完了します

Amazon ECS UI の [Task Definitions] で、[Create New Task Definition] をクリックします。

起動タイプの互換性として [EC2] を選択します。「タスク定義名」を指定し、該当する場合は「タスク・ロール」、「ネットワーク・モード」および「タスク実行ロール」を指定します。

ページの一番下までスクロールし、 [JSON 経由で構成] を選択します。既存のコンテンツをすべて削除し、cssensor-aws-ecs.json ファイルの内容全体をコピーして貼り付けます。

「作成」をクリックして、タスク定義を作成します。作成されると、[タスク定義] の下に一覧表示されます。

次に、[Clusters] に移動し、センサーを展開するクラスター名をクリックします。

「サービス」タブで、「作成」をクリックします。起動タイプとして [EC2] を選択します。上記で作成したタスク定義とそのリビジョンを選択し、クラスターを選択します。[サービス名] と [サービスの種類] を "DAEMON" として指定し、必要に応じて [ネットワーク]、[負荷分散]、および [Auto Scaling] を構成します。情報を確認し、「作成」をクリックして サービスを作成します。作成すると、[サービス]の下に表示されます。サービスの状態が [アクティブ] であることを確認します。[tasks] タブで、すべての ECS コンテナでタスクが実行されていることを確認します。

Amazon ECS クラスターでの Qualys センサーの停止

すべてのコンテナで Qualys コンテナ センサーの実行を停止する場合は、[サービス(Services)] タブからサービスを削除するだけです。これにより、qualys-container-sensor サービスが強制終了されますが、AWS ECS インスタンスからセンサーは削除されません。

永続ストレージなしの起動センサー

ホスト上の永続ストレージを使用せずにセンサーを実行できます。この場合、データはホストに保存されませんが、センサーを基準にした /usr/local/qualys/qpa/data フォルダに保存されます。

永続ストレージなしでセンサーを起動するには、**cssensor-aws-ecs.json** ファイルを変更し、コマンドの追加値として "--sensor-without-persistent-storage" を指定します。

```
"command": [
     "--ecs-mode",
     "--sensor-without-persistent-storage"
],
```

ログを保持するには、"--enable-console-logs" オプションと "--sensor-without-persistent-storage" を併用することをお勧めします。

mountPoints で persistent-volume セクションを削除します。

```
"mountPoints": [
      {
          "sourceVolume": "persistent_volume",
          "containerPath": "/usr/local/qualys/qpa/data"
       },
```

[volumes] で persistent-volume セクションを削除します。

```
"volumes": [
       {
           "name": "persistent_volume",
           "host": {
```

```
            "sourcePath": "/usr/local/qualys/sensor/data"
        }
    },
```

AWS Fargate (ECS) でのコンテナイメージのスキャン

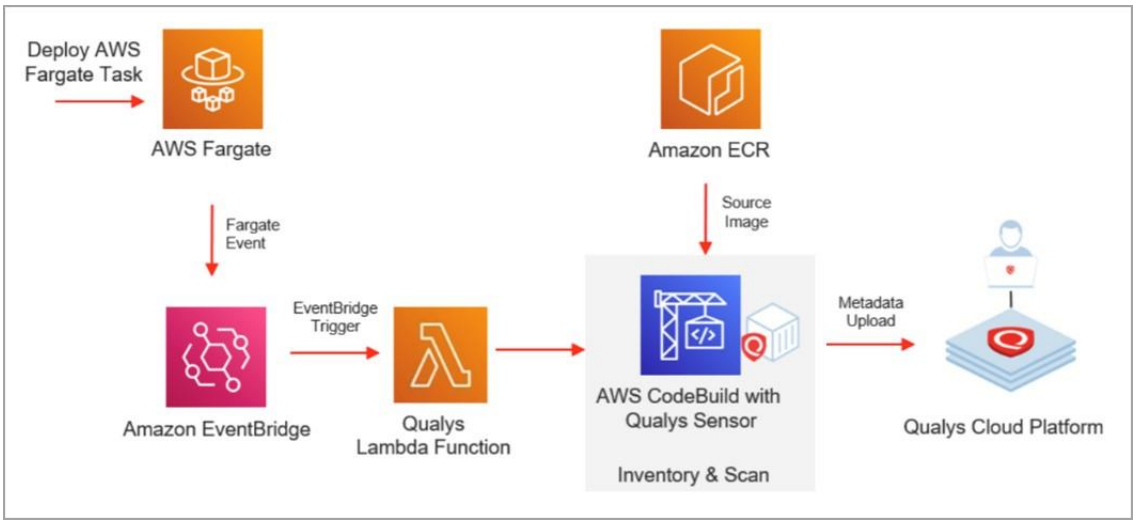
Qualys Container Security を使用して、AWS Fargate を保護できます。AWS Fargate は、Amazon Elastic Container Service (ECS) と連携するコンテナ用のサーバーレスコンピューティングエンジンです。この機能を使用すると、AWS Fargate で実行されているコンテナを把握し、Amazon Fargate タスク (ECS) によって起動されたコンテナイメージに対して脆弱性とコンプライアンスのスキャンを実行し、結果を表示して修復アクションを実行できます。

AWS Fargate はサーバーレスであるため、このソリューションでは、新しい Fargate タスクがデプロイされるたびにセンサーが起動されます。AWS CloudFormation と Qualys Lambda 関数を使用して、スキャンを自動的にトリガーします。サブスクリプションの詳細を使用して CloudFormation テンプレートを設定し、Qualys S3 バケット名と S3 バケットキーを使用して Qualys Lambda 関数を設定し、Amazon Elastic Container Registry (ECR) からプルされたイメージのイメージスキャンをトリガーします。

仕組み

x86_64 アーキテクチャの Amazon Elastic Container Registry (Amazon ECR) からプルされた Docker イメージのスキャンをサポートしています。

AWS ECS Fargate タスクが起動されると、Qualys デプロイ中に作成された AWS EventBridge ルールがイベントを消費します。EventBridge ルールは、Qualys スキャン Lambda 関数をトリガーするように設定されます。その後、Qualys Lambda 関数は EventBridge から受信したイベントを処理して、イメージスキャンを決定します。Qualys Lambda 関数は、AWS CodeBuild を起動して Qualys センサーを実行し、Amazon ECR からイメージをプルし、イメージに対して脆弱性とコンプライアンスのスキャンを実行します。イメージ スキャンが成功すると、イメージ メタデータが評価のために Qualys Cloud Platform にアップロードされ、ユーザーは Container Security UI と API から詳細を表示できます。



Qualys Private Cloud Platform (PCP) については、AWS とプライベート クラウド プラットフォーム間の接続を設定するためのガイドラインを提供しています。「Qualys コンテナのセキュリティ - Qualys プライベートクラウドプラットフォーム(PCP)での AWS Fargate の保護」を参照してください。

Qualys AWS ECS Fargate イメージ スキャン スタックのデプロイ手順

以下の手順に従って、AWS ECS Fargate イメージスキャンを設定します。Qualys CloudFormation テンプレートと Qualys Lambda 関数を使用して CloudFormation スタックを作成します。

前提 条件

開始する前に、CloudFormation AWS ECS Fargate イメージスキャンスタックを正常に起動するために、以下の項目が揃っていることを確認してください。

- スタックをデプロイする AWS リージョン。
- Qualys CloudFormation テンプレート URL: https://qualys-cs-image-scanning-cloudformation-template.s3.amazonaws.com/qcs-ecs-fargate-image-scanningcf.template
- Qualys サブスクリプションの環境の詳細:POD URL、アクティベーション ID、顧客 ID。サブスクリプションのアクティベーション ID とカスタマー ID を自動生成するには、 **UI で** [Configurations > Sensors] に移動し、[Download Sensor] をクリックします。次に、任意のセンサータイプをクリックします。[**インストール手順**] ページのインストール コマンドには、アクティベーション ID とカスタマー ID が含まれています。
- Qualys Lambda 関数 (Zip ファイル)。ECS スキャン Lambda 関数を設定するには、S3 バケット名とバケットキーが必要です。各 AWS リージョンの S3 バケット名とバケットキーを取得するには、Qualys CS Lambda 関数 S3 バケットの名前とキーのセクションを参照してください。
- Qualys センサー イメージ (バージョン 1.18 以降)。『Qualys センサー イメージの取得方法』を参照してください

Qualys センサー イメージの取得方法

Qualys Container Security Sensor イメージには、次のオプションがあります。

- Docker Hub から直接使用する
- Docker Hub から使用しますが、イメージを ECR リポジトリ(パブリック)にプッシュします
- tar からロードし、ECR リポジトリにプッシュする(パブリック)

以下のセクションでは、これらのオプションについて詳しく説明します。

Docker Hub から直接使用する

センサー イメージは、Docker Hub から直接使用できます。Docker Hub の Container Security Sensor は、次のように使用できます。

```
qualys/qcs-senso:<tag>
qualys/qcs-sensor:latest
```

Docker Hub で最新のタグを検索します。

Docker Hub から使用しますが、イメージを ECR リポジトリ(パブリック)にプッシュします

次のコマンドを使用して、qualys センサーイメージを ECR パブリックリポジトリにプッシュします。

```
sudo docker pull qualys/qcs-sensor:latest
sudo docker tag qualys/qcs-sensor:latest <URL to push image to ECR
public repository>
sudo docker push <URL to push image to ECR public repository>
```

例えば：

```
sudo docker pull qualys/qcs-sensor:latest
sudo docker tag c3fa63a818df
public.ecr.aws/y4h7m2t8/qualys/sensor:latest
sudo docker push public.ecr.aws/y4h7m2t8/qualys/sensor:latest
```

tar からロードし、イメージを ECR リポジトリ(パブリック)にプッシュします

Linux コンピュータ上の QualysContainerSensor.tar.xz ファイルを Qualys クラウドポータルからダウンロードします。Container Security UI で、[Configurations > Sensors] > [Download Sensor] に移動してバイナリ(tar.xz)ファイルをダウンロードします。次に、[Linux]と[**バイナリ**(tar.xz)]タブを選択します。 [**今すぐダウンロード**]をクリックして tar ファイルを取得します。

センサーパッケージを解凍します。

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

次のコマンドを使用して、qualys センサーイメージを ECR パブリックリポジトリにプッシュします。

```
sudo docker load -i qualys-sensor.tar
sudo docker tag <IMAGE NAME/ID> <URL to push image to ECR
public repository>
sudo docker push <URL to push image to ECR public repository>
```

例えば：

```
sudo docker load -i qualys-sensor.tar
sudo docker tag c3fa63a818df
public.ecr.aws/y4h7m2t8/qualys/sensor:latest
sudo docker push public.ecr.aws/y4h7m2t8/qualys/sensor:latest
```

AWS コンソールを使用してスタックをデプロイする方法

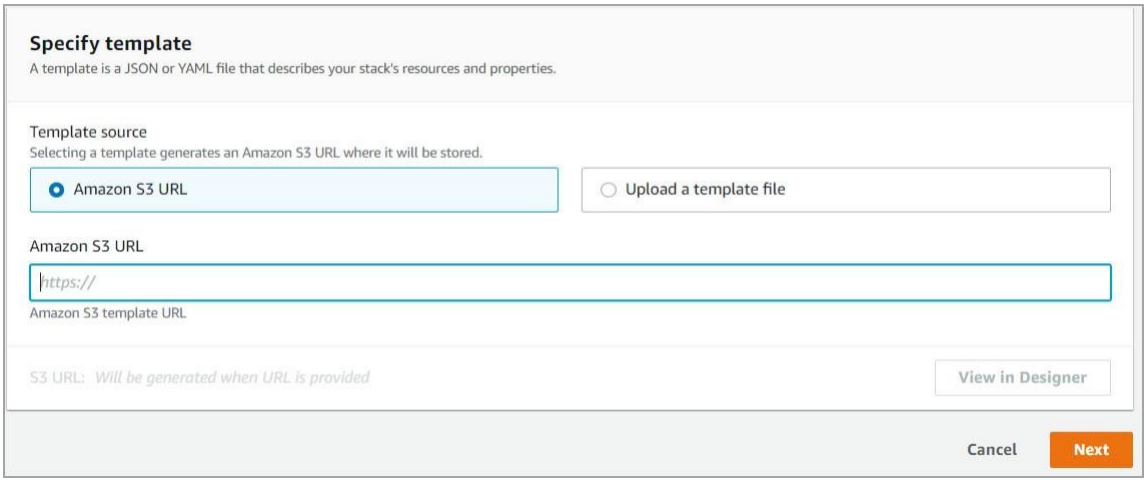
AWS Fargate ECS でのコンテナイメージのスキャンには AWS CloudFormation を使用しています。

次の展開手順に従います。

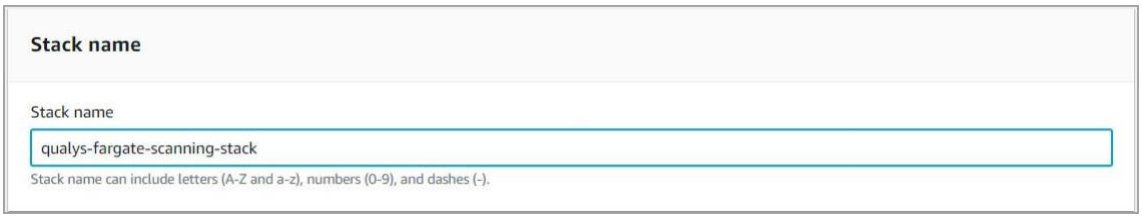
1) AWS コンソールにログインします。

2) CloudFormation に移動し、[Create Stack] をクリックして [With new Resources] を選択します。

3) [テンプレートの指定] の [Amazon S3 URL] フィールドに、Qualys CloudFormation テンプレート S3 URL を入力します(「前提条件」セクションの「URL 」を参照)。次に、「次へ」をクリックして、 テンプレートの構成に進みます。



4) [スタック名] に、Qualys-fargate-scanning-stack などの Qualys AWS Fargate スキャンスタックの名前を入力します。



5) Qualys グローバル設定(Qualys Global Configuration) で、ポッド URL、アクティベーション ID、顧客 ID など、サブスクリプションの環境の詳細を指定します。

サブスクリプションのアクティベーション ID とカスタマー ID を自動生成するには、UI の [Configurations > Sensors] に移動し、[Download Sensor] をクリックしてから、任意のセンサータイプをクリックします。[インストール手順] ページのインストール コマンドには、アクティベーション ID とカスタマー ID が含まれています。

QualysPodUrl:アカウントが配置されている Qualys プラットフォームのコンテナ セキュリティ サーバ URL を入力します(例: US2 プラットフォームの場合は https://cmsqagpublic.qg2.apps.qualys.com/ContainerSensor)。URL がわからない場合は、次のリンクを参照して Qualys プラットフォームを特定し、Container Security Server の URL を取得してください。 https://www.qualys.com/platform-identification/#container-security-servers

QualysCustomerId: Qualys サブスクリプションの顧客 ID を入力します。

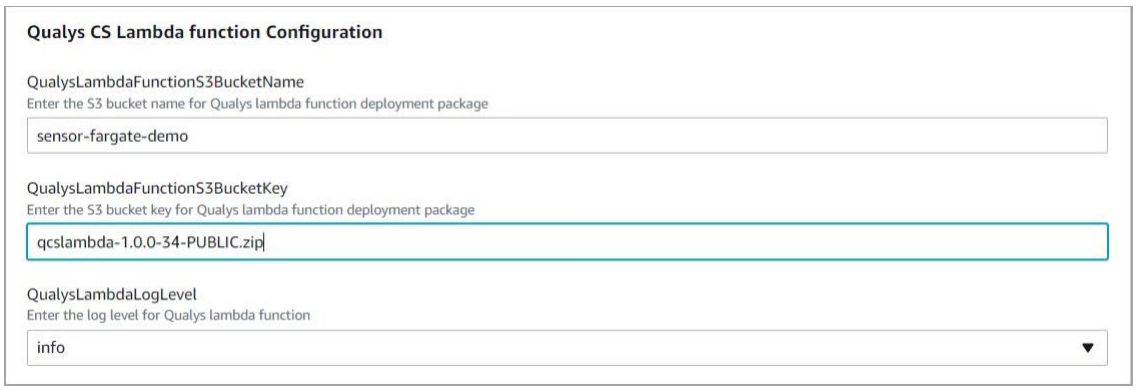
QualysActivationId: Qualys サブスクリプションのアクティベーション ID を入力します。

6) [Qualys CS Lambda 関数の設定] で、Lambda S3 バケット名、キー、ログレベルを指定します。バケット名とキー値を取得するには、次のリンクを参照してください: Qualys CS Lambda 関数 S3 バケットの名前とキー

QualysLambdaFunctionS3BucketName: Qualys Lambda 関数の S3 バケット名を入力します。

QualysLambda 関数 S3 バケットキー: Qualys Lambda 関数の名前として S3 バケットキーを入力します (例: qcslambda-1.0.0-34-PUBLIC.zip)。

QualysLambdaLogLevel: Qualys Lambda 関数のログレベルを選択します。デフォルト値は "info" です。デフォルト値をそのまま使用するか、より詳細なログ記録が必要な場合は別のログレベルを選択します。



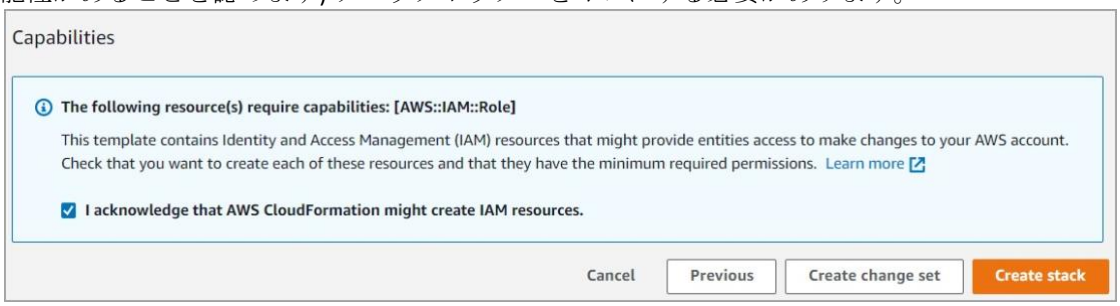
7）Qualys CS **センサー設定(Qualys CS Sensor Configuration)**] で、次の詳細を指定します。

QualysSensorImage: CS 1.18.0 センサー イメージの名前を入力します。

QualysSensorLogLevel: Qualys センサーのログ レベル (0 から 5) を選択します。デフォルト値は 3 (情報) です。デフォルト値をそのまま使用するか、より詳細なログ記録が必要な場合は別のログレベルを選択します。

QualysSensorCLIParameters: このフィールドは空のままにしておくことができます。

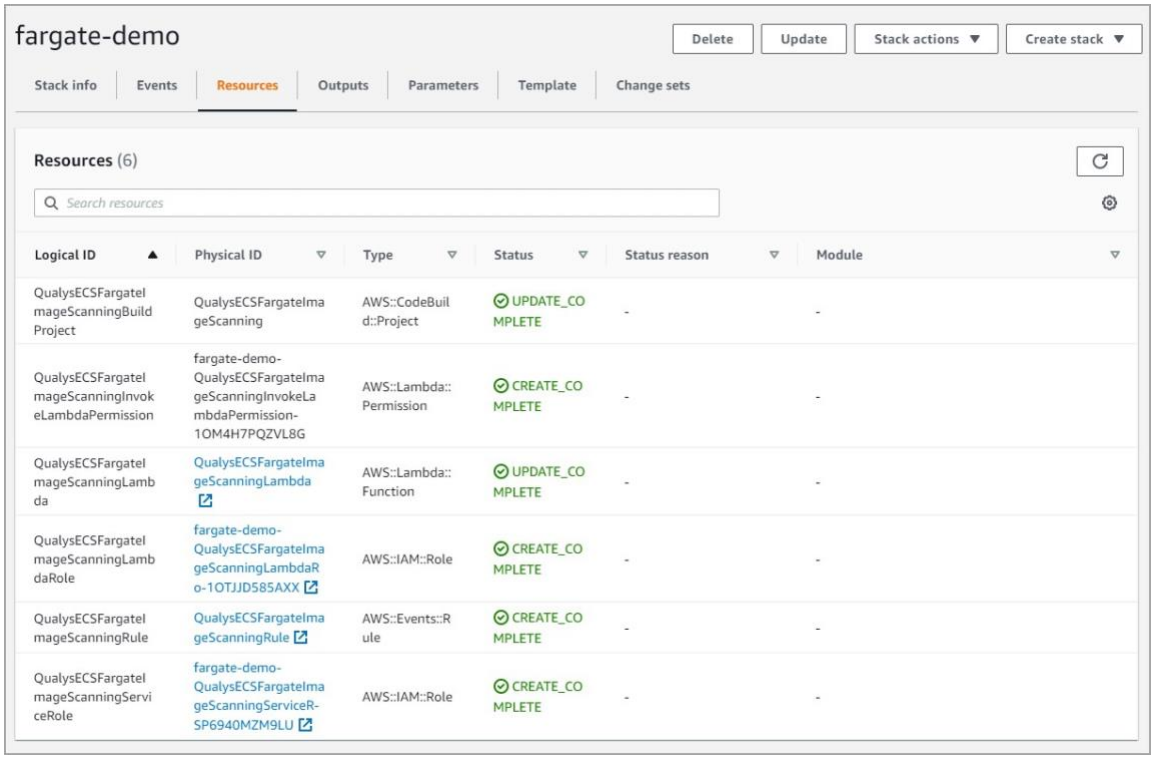
８）「次へ」をクリックして、ワークフローを続行します。最後のページで、[I acknowledge that AWS CloudFormation might create IAM resources] (AWS CloudFormation が IAM リソースを作成する可能性があることを認めます) チェックボックスをオンにする必要があります。



「Create stack」をクリックします。

作成されたリソース

スタックの作成が成功すると、いくつかのリソースが作成され、次に示すように [Resources] (リソース) セクションに表示されます。この例では、リソースは fargate-demo という名前のスタック用に作成されています。



作成されたリソースについて、もう一度見てみましょう。

Logical ID	Type
QualysECSFargateImageScanningBuildProject	AWS::CodeBuild::Project
QualysECSFargateImageScanningInvokeLambdaPermission	AWS::Lambda::Permission
QualysECSFargateImageScanningLambda	AWS::Lambda::Function
QualysECSFargateImageScanningLambdaRole	AWS::IAM::Role
QualysECSFargateImageScanningRule	AWS::Events::Rule
QualysECSFargateImageScanningServiceRole	AWS::IAM::Role

Qualys CS Lambda 関数 S3 バケットの名前とキー

CS Lambda 関数 S3 バケット名

以下の表を参照して、リージョンの Qualys CS Lambda 関数 S3 バケット名を取得してください。

AWS Region	Bucket Name
us-east-1	qualys-cs-image-scanning-lambda-function-us-east-1
us-east-2	qualys-cs-image-scanning-lambda-function-us-east-2
us-west-1	qualys-cs-image-scanning-lambda-function-us-west-1
us-west-2	qualys-cs-image-scanning-lambda-function-us-west-2

me-south-1	qualys-cs-image-scanning-lambda-function-me-south-1
eu-central-1	qualys-cs-image-scanning-lambda-function-eu-central-1
eu-west-1	qualys-cs-image-scanning-lambda-function-eu-west-1
eu-west-2	qualys-cs-image-scanning-lambda-function-eu-west-2
eu-north-1	qualys-cs-image-scanning-lambda-function-eu-north-1

CS Lambda 関数 S3 バケットキー

Qualys CS Lambda 関数の S3 バケットキーは、すべての AWS リージョンで同じです。

入力するバケットキーは **qcslambda-1.0.0-34-PUBLIC.zip** です。

CIS Benchmark for Docker への準拠

Qualys Container Security は、センサー イメージの Docker の CIS ベンチマークに準拠しています。このセクションでは、Docker の CIS ベンチマークに準拠する方法でセンサー イメージを使用する方法に関するガイダンスを提供します。準拠した方法でセンサーを操作できるように、いくつかのコントロールについて以下の手順を提供します。

CIS Docker Benchmark	5.9 Ensure that the host's network namespace is not shared (Automated)
Qualys Control	CID 10811 "Status of the network mode set for the Docker containers on the host system"
Resolution	To meet compliance with this control, Qualys Container Sensor should run without the command line argument --net=host. However, it should be noted that when sensor is launched without --net=host, sensor will not be able to detect its host IP address. By default, the sensor container will be assigned a default IP from the pool assigned to the network. All the containers on the host will be mapped to the IP assigned to the sensor container and each container's host IP association will be missing. Hence, not to lose the asset/host association we recommend the sensor to be launched with the argument.
Installsensor.sh Command	Remove --net=host from the installsensor.sh script to run sensor without this command.
Docker Run Command	Do not specify --net=host as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest
Kubernetes DaemonSet	For deployments in Kubernetes with Docker Runtime, remove hostNetwork: true from cssensor-ds.yml.
CIS Docker Benchmark	5.10 Ensure that the memory usage for containers is limited (Automated)
Qualys Control	CID 10812 "Status of the memory usage limitation for the Docker containers on the host system"

Resolution	To meet compliance with this control, Qualys Container Sensor should run with the command line argument MemoryUsageLimit for the installsensor.sh script or -m as part of Docker run command when deploying a sensor. The value should be formatted as <digit><unit> where unit can be any of the following: b (bytes), k (kilobytes), m (megabytes), g (gigabytes). The recommended value is 500m for 500 megabytes.
Installsensor.sh Command	Specify MemoryUsageLimit as a command line argument for installsensor.sh script. sudo ./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id> Storage=/tmp/qualys/sensor/data MemoryUsageLimit=500m s
Docker Run Command	Specify -m as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure -m 500m -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<CustomerId> -e POD_URL=<POD URL> -net=host --name qualys-container-sensor qualys/qcssensor:latest
Kubernetes DaemonSet	For deployments in Kubernetes with Docker Runtime, add the memory usage limit to the resources section in the cssensords.yml file, as shown below. resources: limits: memory: "500M"
CIS Docker Benchmark	5.11 Ensure that CPU priority is set appropriately on containers (Automated)
Qualys Control	CID 10813 “Status of the CPU share weighting set for the Docker containers on the host system”
Resolution	To meet compliance with this control, Qualys Container Sensor should run with the command line argument CpuShares for the installsensor.sh script or --cpu-shares as part of Docker run command when deploying a sensor. This defines the CPU shares for the sensor container. A valid value is a non-zero, positive integer other than 1024.

Installsensor.sh Command	Specify CpuShares as a command line argument for installsensor.sh script. sudo ./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id> Storage=/tmp/qualys/sensor/data CpuShares=1023 -s

Docker Run Command	Specify --cpu-shares as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure --cpu-shares 1023 -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest
CIS Docker Benchmark	5.12 Ensure that the container's root filesystem is mounted as read only (Automated)
Qualys Control	CID 10825 "Status of the read-only filesystem for the Docker containers on the host system"
Resolution	To meet compliance with this control, Qualys Container Sensor should run with the command line argument --readonly for the installsensor.sh script or as part of Docker run command when deploying a sensor.
Installsensor.sh Command	Specify --read-only as a command line argument for installsensor.sh script. sudo ./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id> Storage=/tmp/qualys/sensor/data --read-only -s
Docker Run Command	Specify --read-only as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure --read-only -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest
Kubernetes DaemonSet	For deployments in Kubernetes with Docker Runtime, add readOnlyRootFilesystem: true in the securityContext section of the cssensor-ds.yml file. securityContext: readOnlyRootFilesystem: true

CIS Docker Benchmark	5.25 Ensure that the container is restricted from acquiring additional privileges (Automated)
Qualys Control	CID 10855 "Status of the 'no-new-privileges' security option set for the Docker containers on the host system"
Resolution	To meet compliance with this control, Qualys Container Sensor should run with the command line argument --security-opt=no-new-privileges as part of the Docker run command when deploying a sensor.

Installsensor.sh Command	There is no new option for the installsensor.sh script. Support for installer script for this argument will be added in a future release. Alternatively, you can use docker run command as specified below to meet this control.
Docker Run Command	Specify --security-opt=no-new-privileges as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure --security-opt=nonew-privileges -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest
CIS Docker Benchmark	5.28 Ensure that the PIDs cgroup limit is used (Automated)
Qualys Control	CID 10829 “Status of the Process ID (PID) cgroup limit for Docker containers”
Resolution	To meet compliance with this control, Qualys Container Sensor should run with the command line argument PidLimit for installsensor.sh script or --pids-limit as part of Docker run command when deploying a sensor. This defines the Pid limit for the sensor container. The value provided must be a positive integer.
Installsensor.sh Command	Specify PidLimit as a command line argument for installsensor.sh script. sudo ./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id> Storage=/tmp/qualys/sensor/data PidLimit=100 -s
Docker Run Command	Specify --pids-limit as part of the Docker run command when deploying a sensor. sudo docker run -d --restart on-failure --pids-limit 100 -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest

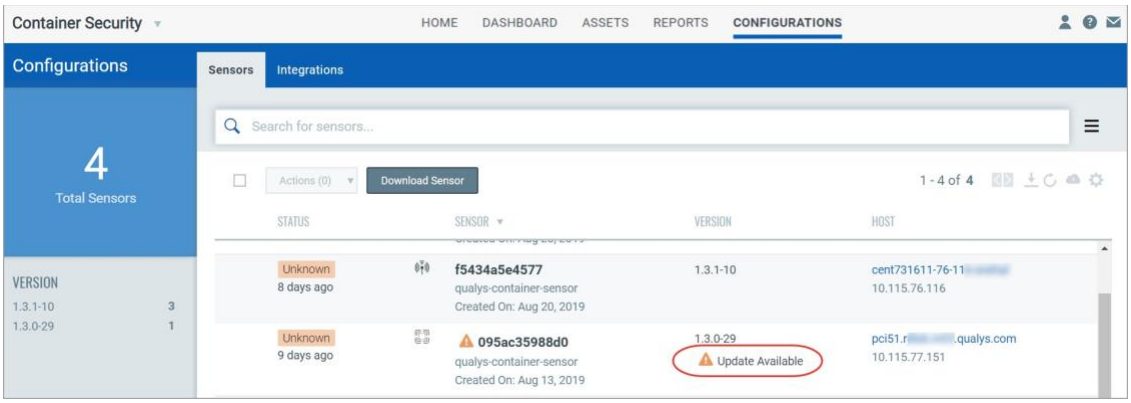
CIS Docker Benchmark	**Multiple Controls**

Installsensor.sh Command	This sample includes all the command line arguments for installsensor.sh for meeting compliance, including MemoryUsageLimit, CpuShares, PidLimit, and --read-only. Note that --net=host is excluded. sudo ./installsensor.sh ActivationId=<Activation id> CustomerId=<Customer id> Storage=/tmp/qualys/sensor/data MemoryUsageLimit=500m CpuShares=1023 PidLimit=100 -s --read-only
Docker Run Command	This sample includes all the command line arguments for the Docker run command for meeting compliance, including --security-opt=no-new-privileges, --cpu-shares, --pids-limit, -m, and --read-only. Note that --net=host is excluded. sudo docker run -d --restart on-failure --read-only --securityopt=no-new-privileges --cpu-shares 1023 --pids-limit 100 -m 500m -v /var/run/docker.sock:/var/run/docker.sock:ro -v /usr/local/qualys/sensor/data:/usr/local/qualys/qpa/data -e ACTIVATIONID=<Activation id> -e CUSTOMERID=<Customer id> -e POD_URL=<POD URL> --name qualys-container-sensor qualys/qcs-sensor:latest

Administration

センサーの更新

デプロイされたものよりも新しいセンサー バージョンが使用可能な場合は、UI のセンサー名の横に [利用可能な更新プログラム] と表示されます。センサーを新しいバージョンに更新して、新機能やバグ修正を活用し、脆弱性を修正する必要があります。



Qualys UI からダウンロードしたセンサーの場合

installsensor.sh script または docker run コマンドを使用して Docker にデプロイされたセンサーは、自動的に更新されます (インストール スクリプトに --disable-auto-update オプションが使用されていない場合)。センサーは、Kubernetes デプロイ用に自動的に更新されません。手順については、「Kubernetes にデプロイされたセンサーを更新する」を参照してください。

Docker Hub からインストールされたセンサーの場合

Docker Hub でホストされている Qualys Container Sensor イメージは、自動更新をサポートしていません。 最新バージョンにアップグレードする方法については、「Docker Hub からのセンサーのインストール」セクションの「センサーのアップグレード」を参照してください。

センサーのアンインストール方法

QualysContainerSensor.tar.xz ファイル(Qualys Cloud Platform からセンサーをインストールするためにダウンロード)には、センサーを**アンインストールするためのスクリプト** uninstallsensor.sh が含まれています。

センサーをアンインストールするには:

Docker ホストが docker.sock 経由で通信するように構成されている場合は、次のコマンドを使用します。

```
./uninstallsensor.sh -s
```

Docker ホストが TCP ソケット経由で通信するように構成されている場合は、Docker デーモンがリッスンするように構成されているアドレスを指定します。

```
./uninstallsensor.sh DockerHost=<<IPv4 address or FQDN>:<Port#>> -s
```

例えば

```
./uninstallsensor.sh DockerHost=10.11.12.13:1234 -s
```

画面の指示に従って、センサーをアンインストールします。Qualys では、永続ストレージをクリアしないことを推奨しています。

トラブルシューティング

センサーのログを確認する

センサー ログ ファイルは、既定で次の場所にあります。

/usr/local/qualys/sensor/data/logs/qpa.log

引数 "--sensorwithout-persistent-storage" を使用して永続ストレージなしでセンサーを実行している場合、センサーはホストにデータを書き込まないことを意味します。この場合、ログはセンサー コンテナ内の次の場所にあります。

/usr/local/qualys/qpa/data/

センサーの正常性状態

Container Security Sensor v1.10 以降のスタンドアロン デプロイの場合、コンテナの正常性ステータスが "docker ps" に表示されます。デプロイから最初の 24 時間は、センサーの正常性を監視するため、正常性状態は "正常性: 開始中" になります。24 時間後、ステータスは "health: starting" から "healthy" に変わります。「docker ps」のセンサーSTATUS が「Up」である限り、センサーは正常に実行されていることに注意してください。

新しいセンサーのデプロイの最初の 24 時間のセンサーの正常性状態の例を次に示します。

```
root@ip-10-11-12-13:/home/ubuntu# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5e234d567c89 qualys/sensor:latest "/usr/bin/tini -- /u…" 2 days ago Up
10 minutes (health: starting) qualys-container-sensor
```

診断スクリプト

Qualys には、センサーに関する診断情報を収集するためのスクリプトが用意されています。このスクリプトは、診断情報の収集元となるホストで実行する必要があります。

診断スクリプトは、センサーをインストールするためにダウンロードした QualysContainerSensor.tar.xz にあります。

スクリプトは Sensor_Diagnostic_Script.py と呼ばれます。スクリプトを実行するには、ホストに Python がインストールされている必要があります。このスクリプトは、ホストから次の情報を収集し、SensorDiagnostic.tar という tar ファイルに格納します。そのファイルを Qualys サポートに送信して、さらにサポートを受けることができます。

SensorDiagnostic.tar には、'ScanInfo.json'、特定の永続ストレージからの qualys-container-sensor の 'qpa.log'、qualys-container-sensor の docker ログ、および 'SensorDiagnostic.log ファイルの以下で説明するすべての情報が含まれます。Docker ホストで 'ScanInfo.json' とセンサー ログが使用できな

い場合、このスクリプトは空の 'ScanInfo.json' ファイルと qpa..log ファイルを作成し、それらに " ファイルが見つかりません" を追加します。

- オペレーティングシステム情報(OS の種類、つまり Linux または Mac、およびその他の詳
- プロキシ構成 (プロキシ セットの種類 (システム、Docker、クラウド エージェント プロキシなど)
- CPU アーキテクチャ(モデル、CPU、コアなどの詳細)
- RAM 使用量 (ホスト上のメモリ割り当てと使用率)
- Docker バージョン(ホストにインストールされている Docker バージョン)
- ソケット構成 (TCP/UNIX ドメインなどのホスト上の Docker ソケット構成)
- Docker イメージの数(すべての Docker イメージとその詳細の数)
- Docker コンテナの数(すべての Docker コンテナとその詳細の数)
- 実行中のコンテナの CPU とメモリの使用率 (すべてのリソース使用状況の統計の最初の結果)

アップグレード中にセンサーがクラッシュする

installsensor.sh を使用して Qualys コンテナ センサーを再インストールし、以前のセンサーの「ストレージ」値を維持します。これにより、新しいセンサーが別のセンサーとしてマークされることはなく、既存のセンサーをアップグレードするだけです。

install コマンドのヘルプについては、「Container Sensor のデプロイ」を参照してください。

注: どの時点でも、永続ストレージを削除しないでください。それ以外の場合、その後に展開されたセンサーは新しいセンサーとしてマークされます。

センサーが再起動した場合はどうなりますか?

センサーは再起動シナリオを処理するように設計されており、再起動後も正常に機能し続けます。センサーがクラッシュするまで、お客様の介入は必要ありません。センサーは、センサーの再起動ポリシーに従って再起動します。

Sensor restart policy

例外は正常に処理され、以下で説明するように、回復可能なエラーと回復不能なエラーの再起動ポリシーに従ってセンサーが再起動します。

Recoverable errors

センサーは、センサーがクラッシュした場合やセンサーが例外をキャッチした場合など、回復可能なエラー コード 24 を返します。このような場合、センサーは自然に回復し、再起動を続けます。再起動の回数に上限は設定されていませんが、2 回の再起動の間隔は、必要な再起動の回数に応じて最大 16 分まで長くなります。たとえば、2 回の再起動の間隔は、開始まで 1 分、2 分、4 分、8 分、16 分と続きます。16 分に達すると、再起動の間隔は 16 分のままです。コア ダンプ ファイルは作成されません。

Irrecoverable errors

センサーが回復不能なエラー コードを返す場合は、センサーがそれ自体で回復せず、センサーが終了することを意味します。スタンドアロン展開の場合、センサーは回復不能なエラー コードを受信すると終了します。DaemonSet デプロイの場合、センサーが回復不能なエラー コードで終了すると、Kubernetes ポッドの再起動ポリシーによって、終了したコンテナが再起動されます。回復不能なエラー コードは、配置ファイルと配置引数を変更して解決する必要があります。

Kubernetes コンテナの複製

コンテナの検索中に、Kubernetes によってオーケストレーションされたコンテナの重複が表示される場合があります。これは、Kubernetes が起動するすべてのサービスコンテナに対して監視コンテナを起動するためです。Qualys コンテナ センサーは、これらを 2 つの異なるコンテナとして認識し、両方のコンテナを報告してスキャンします。

コンテナが重複していない結果を表示するには、Kubernetes コンテナの検索に使用するクエリに次の文字列を追加します。

```
not label.key:POD
```

たとえば、次のクエリを使用して、Kubernetes で実行中のコンテナを検索します。

```
state:"RUNNING" and not label.key:POD
```

コンテナのランタイムの詳細を取得する

使用中のコンテナランタイムと設定の詳細を取得するために実行できるコマンドがいくつかありますが、問題のトラブルシューティングのために Qualys サポートと共有することができます。

Get nodes

次のコマンドを使用して、名前、状態、ロール、経過時間、バージョン、内部および外部 IP アドレス、OS イメージ、カーネル バージョン、コンテナ ランタイムなど、各ノードの詳細を含むクラスターに関する情報を取得します。

```
kubectl get nodes -o wide
```

Get container runtime info

次のコマンドを使用して、ステータスや構成などのコンテナランタイムに関する情報を取得します。

```
crictl info
```

List containers

次のコマンドを使用して、コンテナ ID、イメージ ID、コンテナが作成された日時 (分数、日数、週数、または月数)、現在の状態 (実行中、終了など)、コンテナ名、試行回数、POD ID などの詳細を含むすべてのコンテナを一覧表示します。

```
crictl ps -a
```

List images

次のコマンドを使用して、イメージ名、タグ、イメージ ID、イメージ サイズなどの詳細を含むイメージを一覧表示します。

```
crictl images
```